# A LIGHTWEIGHT MOBILE CLOUD COMPUTING FRAMEWORK FOR RESOURCE-INTENSIVE MOBILE APPLICATION

SAEID ABOLFAZLI TORGHABEH

Faculty of Computer Science and Information Technology
University of Malaya
Kuala Lumpur

2014

# A LIGHTWEIGHT MOBILE CLOUD COMPUTING FRAMEWORK FOR RESOURCE-INTENSIVE MOBILE APPLICATION

SAEID ABOLFAZLI TORGHABEH

THESIS SUBMITTED IN FULFILMENT OF
THE REQUIREMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Faculty of Computer Science and Information Technology
University of Malaya
Kuala Lumpur

2014

# UNIVERSITI MALAYA

## ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Saeid Abolfazli Torghabeh (Passport No.:H95659183)

Registration/Matrix No.: WHA100050

Name of Degree: Doctor of Philosophy

Title of Thesis: A Lightweight Mobile Cloud Computing Framework for Resource-intensive Mobile Application

Field of Study: Distributed Mobile Computing

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

## ABSTRACT

Resource-intensive Mobile Application (RMA) execution is inhibited by mobile device constrained resources, particularly CPU, RAM, storage, and battery. However, Mobile Cloud Computing (MCC) as the state-of-the-art mobile computing paradigm is aiming to augment computing capabilities of mobile devices, mitigate their resource-deficiency, and realize efficient execution of RMA. MCC solutions dominantly perform remote execution of resource-intensive RMAs' components using resources-rich Distant Immobile Cloud (DIC), particularly public cloud. Although DICs feature high availability and elastic scalability, they are characterized by high communication latency and lack of mobility. Therefore, performance gains of mobile augmentation using DIC are mitigated and RMA execution efficiency is remarkably degraded. In this study, we aim to achieve efficient execution of RMAs by proposing a lightweight MCC framework. We verify the problem significance by analyzing time and energy overheads of exploiting DICs for augmenting resource-constraint mobile devices. Results of our analysis unveil that communication latency of utilizing DICs due to manifold intermediate hops between mobile device and DICs significantly prolongs application execution time and expedites energy dissipation in resource-constraint mobile devices. To address the problem, we propose a lightweight MCC framework that enables usage of multitude of proximate resource-rich mobile devices that can provide computing services to the mobile users in vicinity. The proposed framework is evaluated using benchmarking experiments and validated using statistical modeling. The evaluation results advocate that leveraging our proposed framework can substantially reduce RMAs' execution time up to 91.4% and conserve energy of resource-constraint mobile device as significant as 81%.

# ABSTRAK

Perlaksanaan Aplikasi Mobile berintensifkan Sumber (AMS) atau Resource-intensive Mobile Application (RMA) adalah dihalang oleh kekurangan pada peranti telefon mudah alih, terutamanya CPU, RAM, kapasiti penyimpanan data, dan bateri. Walau bagaimanapun, Perkomputeran Awan Mudah alih (PAM) atau Mobile Cloud Computing (MCC) yang merupakan paradigma pada telefon mudah alih terkini adalah mensasarkan untuk; menambah keupayaan pengkomputeran pada peranti telefon mudah alih, mengurangkan kecacatan sumber, dan merealisasikan pelaksanaan AMS atau RMA . PAM atau MCC telah memberi penyelesaian dengan keupayaannya untuk beroperasi dari jarak jauh iaitu dengan pengunaan komponen AMS atau RMA yang dilengkapi oleh Jarak Ketidakbolehgerakan Awan (JKA) atau Distant Immobile Cloud (DIC), terutamanya pada perisian public cloud. Walaupun ciri-ciri DIC mudah diperolehi dan mempunyai kebolehan untuk digunakan dalam skala yang anjal, tetapi ia dikatogerikan sebagai mempunyai sistem komunikasi yang komplikasi dan kekurangan kebolehgerakan. Oleh itu, peningkatan prestasi telefon mudah alih dengan mengunakan JKA atau DIC adalah tersekat dan efisiensi perlaksanaan AMS atau RMA akan menjadi amat lemah. Dalam kajian ini, kita menyasarkan untuk mencapai perlaksanaan AMS atau RMA yang efisien iaitu dengan mencadangkan rangka kerja PAM atau MCC yang ringkas. Kami mengesahkan kebenaran masalah yang dihadapi dengan meganalisis masa dan tenaga terlebih dahulu sebelum mengeksploitasi JKA atau DIC untuk meningkatkan prestasi telefon mudah alih. Hasil daripada analisis mendedahkan komplikasi sistem komunikasi dalam menggunakan JKA atau DIC adalah disebabkan oleh hop pengantara yang banyak diantara telefon mudah alih dan JKA atau DIC yang telah memanjangkan masa perlaksanaan aplikasi serta mempercepatkan pelesapan tenaga dalam peranti mudah alih. Bagi menangani masalah ini, kami mencadangkan

satu rangka kerja PAM atau MCC ringan yang membolehkan penggunaan pelbagai proksimat peranti mudah alih canggih yang boleh memberikan perkhidmatan pengkomputeran kepada pengguna mudah alih di sekeliling. Rangka kerja yang dicadangkan ini dinilaikan menggunakan pengujian tanda aras dan hasil kajian ini disahkan dengan menggunakan pemodelan statistik . Hasil penilaian yang diperolehi menyokong bahawa rangka kerja yang dicadangkan oleh kami boleh mengurangkan masa pelaksanaan RMA sehingga 81% serta boleh memelihara tenaga peranti mudah alih sebanyak 91.4%.

## ACKNOWLEDGEMENTS

There are a number of people without whom this thesis might not have been written, and to whom I am immensely grateful.

I would like to express my sincere gratitude to my advisor Prof. Abdullah Gani for the continuous support of my Ph.D. study and research. His guidance helped me producing a valuable piece of research reported in this thesis.

Innumerable appreciation to my wife, Zohreh, who has been an endless source of encouragement, inspiration, and support. While she has been pursuing PhD, she has always been offering tireless continuous helps and supports in every oppressive and repressive moments. Her working hands and praying lips have been behind every success I achieved in this journey. Special thanks to my mother, parent-in-law, and families who sacrificed their joyful companion and accepted gap and distance for the years I was pursuing Ph.D. Without their emotional support, I could not complete this program.

Moreover, I would like to thank the financial support and assistance by Ministry of Higher Education, Malaysia for the entire period of my PhD. Special thanks to my friend M. Afiq Alamar for helping me in translating documents to in Bahasa Melayu. Last but not least, great thanks and appreciation to Almighty; the Divine who keeps showering endless blessings on me and making impossible possible. This work is dedicated to all our journeys in learning to thrive.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# LIST OF SYMBOLS AND ACRONYMS

| | |
|---|---|
| 4G | 4th Generation. |
| ACH | Asynchronous Communication Handler. |
| CMA | Cloud-based Mobile Augmentation. |
| CPU | Central Processing Unit. |
| CRC | Cyclic Redundancy Check. |
| CRM | Customer Relation Management. |
| CWnd | Congestion Window. |
| DIC | Distant Immobile Cloud. |
| DSL | Domain Specific Language. |
| DVMS | Dynamic VM Synthesis. |
| GPU | Graphical Processing Unit. |
| GUI | Graphical User Interface. |
| IMSI | International Mobile Subscriber Identity. |
| IP | Internet Protocol. |
| LAI | Location Area Identifier. |
| LCD | Liquid Crystal Display. |
| MCC | Mobile Cloud Computing. |
| ME2 | Mobile Empowering Engine. |
| MNO | Mobile Network Operators. |
| MSC | Mobile Service Consumer. |
| MSP | Mobile Service Provider. |
| MTU | Maximum Transmission Unit. |
| OCR | Optical Character Recognition. |
| QoS | Quality of Service. |
| RAM | Random Access Memory. |
| REST | Representational State Transfer. |
| RISC | Reduced Instruction Set Computing. |
| RMA | Resource-intensive Mobile Application. |
| ROA | Resource-Oriented Architecture. |
| SLA | Service Level Agreement. |
| SOAP | Simple Object Access Protocol. |
| SPDE | Service Provider Discovery Engine. |
| TCP | Transmission Control Protocol. |
| TSG | Trusted Service Governor. |
| UDDI | Universal Description Discovery and Integrity. |
| URI | Unified Resource Identifier. |
| VLR | Visitor Location Register. |
| VM | Virtual Machine. |
| VMM | Virtual Machine Manager. |
| VPU | Virtual Processing Unit. |
| WAN | Wide Area Network. |
| Wi-Fi | Wireless Fidelity. |
| WLAN | Wireless Local Area Network. |

# LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

This chapter presents an overview of the research carried out in this thesis. We present motivation in undertaking the research in this thesis and state the research problem that is investigated and addressed in this research. Our research aim and objectives also are presented in this chapter. Furthermore, the research methodology that is proposed to address the research problem is described.

The remainder of this chapter is as follows. Section 1.1 presents the motivation of research followed by Section 1.2 that presents the identified and established research problem. We state the research aim and objectives in Section 1.3 and describe our proposed methodology to address the research problem in Section 1.4. Finally, Section 1.7 presents the layout of this thesis.

## 1.1 Motivation

The emerging trend of cloud-connected mobile computing and three motives encourage the research undertaken in this thesis that are explained as follows.

• **Emerging Trend:** According to Cisco, there exist nearly seven billion mobile devices in the market (*Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018*, 2013) and they have obtained momentous ground as the predominant computing devices in various computing-intensive domains such as multimedia, image processing, and enterprise applications towards surpassing desktop computers (Albanesius, 2011). However, restrained computing resources of mobile devices encumber execution of Resource-intensive Mobile Applications (RMAs) (Sharifi, Kafaie, & Kashefi,

2011). The research in Mobile Cloud Computing (MCC) aims to alleviate resource poverty in mobile devices towards efficient execution of RMAs.

MCC is a nascent and one of the most rapidly emerging computing disciplines that is recognized as the first technological trend of 2014 and 2015 by IEEE Computer Society (*Top Technology Trends for 2014*, 2014), and Gartner (*Gartner Identifies the Top 10 Strategic Technology Trends for 2015*, 2014). According to Google Trends results from Jan 2009 till Oct 2014 depicted in Figure 1.1, MCC is one of the trendiest research domains that is remarkably emerging in a fast pace. Such popularity and emerging trend heralds the need for research and development of this domain and its potential to contribute to the body of knowledge, industrial products and solutions, and quality of human life.

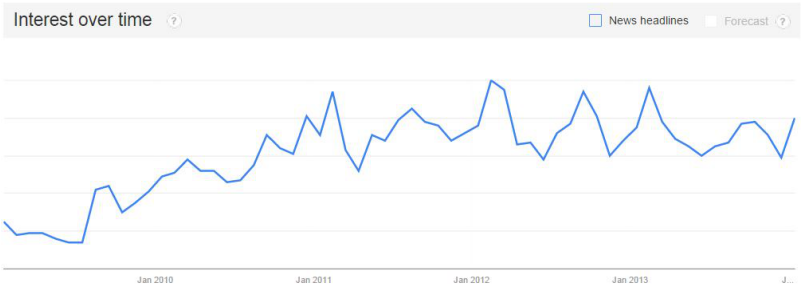



Figure 1.1: Emerging Mobile Cloud Computing Trend From Jan 2009 til Oct 2014 According to Google Trends

• **Contribution to The Body of Knowledge:** Research in mobile augmentation to alleviating shortcomings of mobile devices using computing power of cloud-based resources compliments previous research efforts in load balancing (Othman & Hailes, 1998), power management (Kremer, Hicks, & Rehg, 2003), and computation offloading (Li, Wang, & Xu, 2001) domains which dated back to 90's. Research over application of clouds in mobile computing that is emerging can significantly contribute to the body of knowledge and advance the state-of-the-art mobile and pervasive computing.

• **Advancement of Industrial Products and Solutions:** Results achieved from research and development in alleviating resource poverty of mobile devices enable industri-

alists to advance their products and solutions by developing novel RiMAs that could not exist on normal mobile devices. For instance, augmented mobile device with rich computing capability and low latency can perform live data acquisition and analytics which are critical in several domains, including remote monitoring and operation. Augmented mobile device in plantation farms, for instance, can continuously monitor the environment (the weather, land, and pests) and perform complex decision making algorithms (that cannot be performed on unaugmented mobile devices) to ignite relevant actuators and perform certain tasks, like irrigation.

• **Development of Quality of Human Life:** Human dependency to the contemporary smartphones is rapidly increasing in various domains such as enterprise (Hariharan, 2008), e-learning (Caballe, Xhafa, & Barolli, 2010) and entertainment (Chang, Kwon, & Kang, 2010) due to their unique characteristics, especially miniature nature, handy style, and ubiquity.

Leveraging recent technological advancements, specifically cloud computing and latest achievements in wireless communication (i.e., 4th Generation (4G), Wireless Fidelity (Wi-Fi) hotspot, and IEEE 802.11ah (Hazmi, Rinne, & Valkama, 2012)) to augment mobile devices and alleviate their resource scarcity is crucial to their success and adoption. Although researchers in MCC (Cuervo et al., 2010; Chun, Ihm, Maniatis, Naik, & Patti, 2011; Zhang, Kunjithapatham, Jeong, & Gibbs, 2011a) could enhance computing power of mobile devices and partially alleviate current shortcomings of mobile devices, leveraging distant clouds in these works originates long Wide Area Network (WAN) latency that noticeably increases application execution time and drains the battery (Sharifi et al., 2011; Shiraz, Gani, Hafeez Khokhar, & Buyya, 2012) leading to sharp user experience degradation.

Therefore, proposing a lightweight solution that can mitigate the impact of long WAN

latency in utilizing distant remote resources is a crucial need to empower computing capabilities of resource-constraint mobile devices, which is motivating us to undertaking this research work.

## 1.2 Statement of Problem

*In Mobile Cloud Computing (MCC), empowering computing capabilities of mobile devices, especially smartphones and fulfilling required computational resources of Resource-intensive Mobile Applications (RMAs) are typically undertaken by leveraging rich computing resources of distant immobile clouds (i.e., public clouds). Although distant immobile clouds feature high availability and elastic scalability, performance gain of utilizing such resources is sharply decreased by high communication latency due to large number of intermediate hops between the mobile device and the distant clouds, and RMA execution efficiency is remarkably deteriorated. Therefore, responsiveness and energy-efficiency of RMAs using distant immobile clouds are degraded.*

Local execution of resource-intensive computations of RMAs on mobile devices via native resources either is impossible or leads to immediate battery drainage due to native resource incapacitation. Thus, empowering computing capabilities of mobile devices becomes vital necessity to realize uninterruptible execution of resource-intensive computations on mobile devices without immediate battery drainage, which is possible by exploiting rich computing power of remote cloud-based resources.

RMAs are mobile applications that require intensive computational resources, particularly Central Processing Unit (CPU), Random Access Memory (RAM), storage, and battery to complete the expected computing operation. For instance, image processing applications, enterprise systems, 3-D rendering applications, and video editing applications are exemplary RMAs. Functionalities and operations of RMAs are currently limited due

to resource poverty of mobile devices. Execution of currently available RMAs immediately drains the mobile battery that significantly degrades the quality of user interaction. Hence, mobile resource augmentation become necessary.

In typical mobile empowerment solutions, the resource-intensive component(s) of the RMA are identified, partitioned, and offloaded to the distant cloud datacenters for execution. Upon completion of remote computations, the results are sent back to the mobile device and reintegrated to the rest of the application. Hence, utilizing cloud resources is not a straight forward panacea and is associated with deployment implication and overhead.

Therefore, it is essential to mitigate the overhead of utilizing remote computing resources in smartphone empowerment to avoid jeopardizing the performance gains. It is noteworthy that execution of non-resource-intensive mobile applications is typically undertaken using native resources of the host mobile device. In the other word, exploiting remote cloud-based computing resources is not designed for non-resource-intensive mobile applications, though it is feasible and may be beneficial.

However, though cloud datacenters are large cluster of computing resources with high elastic scalability, utilizing their resources is associated with long WAN latency due to numerous intermediate hops. Distant immobile clouds are usually located in limited geographical regions (e.g., Amazon EC2 as a well-known cloud service provider has datacenters in only 9 regions worldwide) which are far from majority of the mobile users. Moreover, cloud datacenters are immobile computing infrastructures that cannot be momentously migrated from one geographical region to another region to reduce the number of intermediate hops and mitigate the long WAN latency for each mobile device. Also, it is financially very expensive, economically unfeasible, and also insecure (if technologically is possible) to establish a high throughput one-hop communication link from each

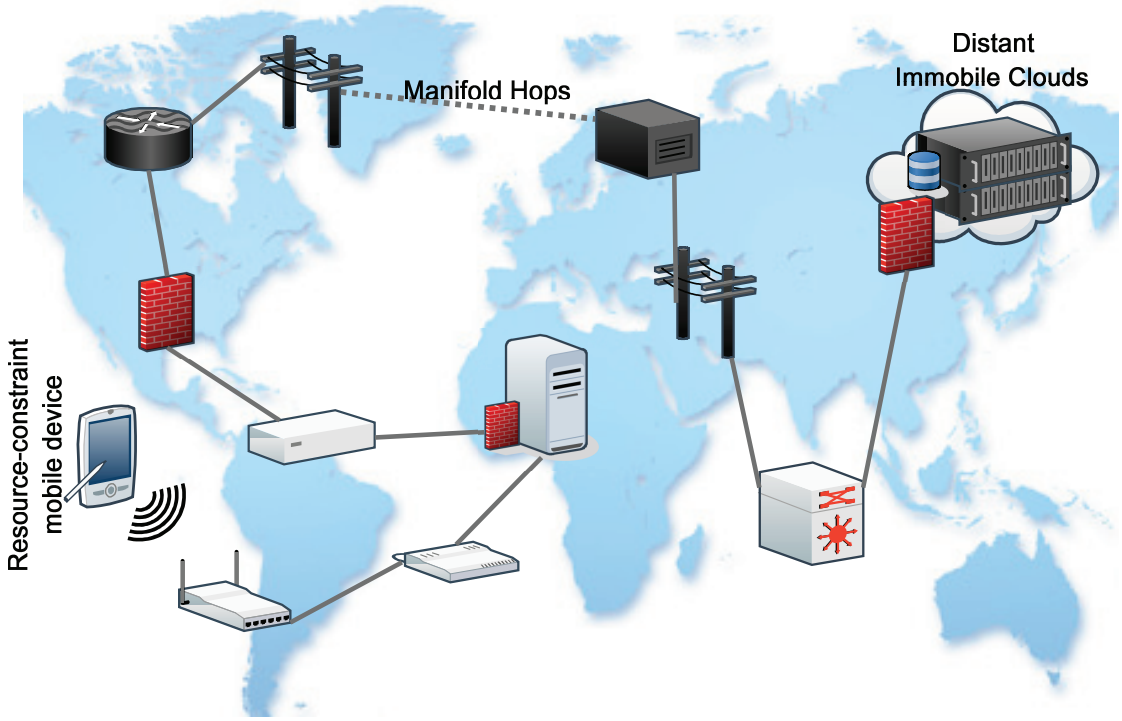



Figure 1.2: High communication latency due to large number of intermediate hops when distant immobile clouds are used to empower resource-constraint mobile devices.

mobile device to the cloud datacenters. Therefore, as illustrated in Figure 1.2, accessing computing resources of distant immobile cloud datacenters originates long WAN latency while traveling through the intermediate hops.

Long WAN latency adversely impacts on the application execution time and prolongs the runtime, because execution of the intensive components of the RMAs is associated with long communication latency of migrating contents (i.e., data and codes) to the distant immobile clouds. Prolonged execution time of the RMAs leads to consumption of more native resources, encumbers usability of applications, and degrades user experience. Moreover, increase in execution time of mobile applications increases energy consumption of the mobile devices and quickly drains the limited battery of the mobile device. There-

fore, due to high communication overhead, utilizing distant immobile cloud datacenters significantly degrades the efficiency of RMAs execution, which necessitates undertaking of further research.

Efficiency in our framework is considered from two aspects of execution time and energy. For time efficiency, we aim to reduce the amount of time required to complete execution of the RMA. Similarly, for energy efficiency, we aim at decreasing the energy requirement for execution of the RMA. For instance, if execution of a RMA takes 100 ms time and 100 mJ energy to complete, we achieved 80 % time and energy efficiency if our proposed solution causes the same RMA executes in 20 ms and consumes 20 mJ energy.

### 1.3 Statement of Objectives

In this research, we aim to achieve efficient execution of compute-intensive mobile applications (as one of RMA types) in resource-constraint mobile environment by mitigating the overhead of performing time- and energy-intensive components in proximate remote cloud-based computing resources. We seek to undertake following steps to achieve our aim.

- Study the research undertaken on RMAs, highlight the deficiencies, and identify the most significant deficiency for alleviation in this research.
- Investigate the identified RMAs' execution inefficiency to demonstrate its significance and establish it as the research problem of this research.
- Design and implement a lightweight MCC framework for efficient execution of RMAs that
    1. Reduces execution time of performing compute-intensive tasks
    2. Decreases energy consumption of performing compute-intensive tasks

- Evaluate the execution performance of the proposed lightweight MCC framework from two views of application execution time and energy consumption via benchmarking on android-based smartphone.

- Validate the results of performance evaluation of the framework based on execution time and energy consumption using statistical modeling.

### 1.4 Proposed Research Methodology

We review the latest credible research efforts to gain insight into the RMA execution domain and determine the significant weaknesses and shortcomings of the recent mobile empowerment approaches. We review recent literature collected from online scholarly databases, particularly IEEE, ACM, Elsevier, and Web of Science to identify inefficiencies of RMA's execution and identify the most critical inefficiency to address in this research. We analytically analyze the identified inefficiency to demonstrate its significance on efficient RMA's execution. Using primary data extracted via benchmarking, the results of analytical analysis are validated and significance of the research problem is demonstrated.

Benchmarking is"the process of performance comparison for two or more systems by measurements" (Jain, 2008) and the benchmarks are "the workloads used in the measurements" (Jain, 2008). "When comparing two or more programs designed to do the same set of tasks, it is customary to develop a small collection of typical inputs that can serve as benchmarks. That is, we agree to accept the benchmark inputs as representative of the job mix; a program that performs well on the benchmark inputs is assumed to perform well on all inputs" (Aho & Ullman, 1992).Therefore, we select workloads as benchmarks to undertake benchmarking experiments. Workloads are input values given to the under study system(s) used in the benchmarking process to indicate the amount of work to be performed by the system.

To address the research problem and achieve the research objectives, we propose a lightweight MCC framework that aims to mitigate the network overhead of performing time- and energy-intensive components of RMAs outside the mobile device. In this framework, we leverage computing capabilities of multitude of proximate mobile devices to perform time- and energy-intensive computations on behalf of nearby resource-constraint mobile devices.

To evaluate the proposal, we devise several series of experiments on real test-bed. In the first series of experiments, we used Javascript and PHP to build a RMAs consists of three compute-intensive computing services: namely prime verification, matrix covariance, and matrix multiplication, and use benchmarking method on android-based smartphones to systematically measure the execution time and energy consumption of the RMAs using 30 workloads when running using the proposed framework. Prime and Matrix are used for two main reasons. Firstly, prime and matrix are mathematical functions that are highly used in standard and popular benchmarks in the literature (Jain, 2008). Prime is the only operation in the Sieve benchmark and matrix operation is the core of LINPACK benchmark which are among popular/standard benchmarking algorithms (Jain, 2008). Secondly, these operations, originate CPU computations that are highly used in real benchmarks (Curnow, Harold J. & Wichmann, 1976). For example, these operations are frequently used in OCR, LNP, image processing, voice processing, face recognition, fingerprint recognition, games, and education purpose. We synthesize the time and energy results of execution using our framework with the results of native execution in the mobile device. Moreover, we build a statistical model to validate the results of performance evaluation. The statistical model is generated using linear regression model which is a predominant observation-based modeling method. The statistical model is validated using split-sample validation approach.

For demonstrating that the performance of our proposed framework does not depend on a certain programming language, mobile device or even particular benchmarking algorithm, we devise second set of experiments. Sieve of Eratosthene is selected as an standard benchmarking algorithm. For implementation, ASP programming language is identified. The real test-bed is designed using Windows-based mobile devices that are running benchmarks over Sieve algorithm. The performance evaluation is undertaken using ten benchmarks, because performance evaluation of computing systems using Sieve benchmark is proven to be sufficient with ten benchmarks only (Jain, 2008).

The third series of experiment, is the comparative study by which we demonstrate the performance of our proposed framework in comparison with related frameworks, particularly Cloudlet (Satyanarayanan, Bahl, Caceres, & Davies, 2009).

## 1.5 Scope

The research undertaken in this thesis is focusing on compute-intensive RMAs only which require intensive computational resources, including CPU, RAM, and battery (Kumar & Lu, 2010). In this study, we do not consider other two types of RMAs (Kumar & Lu, 2010), including data-intensive and communication-intensive RMAs.

Data-intensive applications require highly voluminous data transfer over the wireless network that significantly impacts on efficiency of offloading approaches (Kumar & Lu, 2010). Similarly, communication-intensive applications such as interactive applications that requires continuous interaction (data entry or context acquisition from user or mobile environment) also are not considered in this thesis, because the overhead of frequent communications between mobile and remote servers is likely degrading the performance gain of offloading (Kumar & Lu, 2010).

In this research, we use smartphone and mobile devices interchangeably with identi-

cal notion.

## 1.6 Limitations

The work reported in this thesis is limited from the following aspects:

- The mobile operating system is limited to Android only because (i) it is the most popular mobile operating system, (ii) Android is open source in nature and is highly used in research (Siegal, 2014).

- The smartphone used in the entire experiments in this thesis is a HTC Nexus One that features a RISC 32-bit Qualcomm Snapdragon S1 QSD8250 chipset 1 . This chip has a single core ARMv7 application processor with maximum 1024 MHz clock frequency. The main reasons for selecting such mobile device are that it firstly represent a wide range of mobile devices and it is neither too resource-full and nor too resource-poor. Secondly, the PowerTutor 1.4 that is highly used in energy data collection in MCC works is designed for this mobile device and another two similar devices (according to the developers (*PowerTutor:A Power Monitor for Android-Based Mobile Platforms*, n.d.)). PowerTutor is capable of accurately collecting energy data from this device. Though this application can be executed on other devices such as Samsung Galaxy II and HTC One X (we have experimented), the energy collection is incomplete and application fails to collect wireless communication data.

## 1.7 Thesis Layout

The layout of the thesis including seven chapters is illustrated in Figure 1.3 and the rationals for contents of each chapter are briefly summarized in Table 1.1. For the sake of

brevity and simplicity, headings are condensed. The remainder of this thesis is organized as follows.

Chapter 2 reports a comprehensive review of the state-of-the-art research from literature and identifies the open research problems. It also presents an overview of RMAs, highlights their major requirements and challenges that mitigate RMAs efficiency and hinder successful deployment and development of RMAs. The most credible state-of-the-art research efforts aiming to empower mobile devices and applications are analyzed and synthesized to devise a taxonomy. The comparison of the recent efforts is presented too. Furthermore, the chapter extracts several open research issues and identifies the most significant problem to be addressed in this thesis.

In chapter 3, we aim to analytically and experimentally demonstrate the significance of the identified problem. We analyze the performance of executing RMAs in distant immobile cloud resources to investigate the impact and significance of the long WAN latency stemmed from large number of intermediate hops between smartphone and the distant immobile cloud resources. Using analytical analysis we derive mathematical equations to identify the contributing time-consuming components in execution of RMAs to demonstrate the significant of WAN latency when utilizing remote resources. The findings of this analysis are verified via benchmarking experiments in real MCC environment including android-based mobile device and Amazon EC2 cloud Virtual Machine (VM) instances.

Our proposed framework is described in chapter 4. Using schematic presentation, we present the major components of the proposed framework and describe their functionalities in detail. Coordination of major building blocks is described and illustrated using sequential diagram. Moreover, data design used for performance evaluation is discussed.

Chapter 5 presents the performance evaluation methodology. In this study, we describe how to evaluate the performance of our proposed framework using series of bench-

Table 1.1: Summary of Chapters & Contents and Rational of the Contents Presented in This Thesis

| Chapter | | Contents | Rational |
|---|---|---|---|
| Chapter 1 | Introduction | Motivation | To state the rational for undertaking the current work |
| | | Statement of Problem | To state the research problem identified for alleviation |
| | | Statement of Objectives | To state the aim of the thesis and objectives to attain the aim |
| | | Research Methodology | To state the steps taken to achieve the aim and objectives |
| | | Thesis Layout | To demonstrate the structure of contents presented in thesis |
| Chapter 2 | Literature Review | Resource-intensive Mobile Applications (RMA) | To introduce the RMAs and their characteristics |
| | | RMA Challenges | To identify development and adoption challenges of RMAs |
| | | Taxonomy of Cloud-based Resources | To identify four types of cloud-based resources in MCC |
| | | Review of mobile empowerment approaches | To identify drawback & limitations of current works in literature |
| | | Open Challenges | To highlight identified challenges as future research directions |
| Chapter 3 | Problem Analysis | Analytical latency analysis | To analytically investigate impact of intermediate hops on WAN latency in MCC |
| | | Benchmarking | To empirically analyze impact of intermediate hops on RMA execution efficiency |
| Chapter 4 | Proposed Framework | Schematic Presentation | To schematically present our framework |
| | | Building Blocks | To introduce core blocks of the proposed framework |
| | | Significance | To state the significance & novelty of the proposed framework |
| | | Data Design | To describe data generation methods to evaluate our framework |
| | | Statistical Methods | To describe the statistical methods used to analyze and synthesize the performance analysis results |
| Chapter 5 | Evaluation | Benchmarking Modeling | To describe specification of the model designed for evaluating the performance of the proposed framework |
| | | Statistical Modeling | To describe steps taken to produce the statistical model that aims to validate the evaluation findings<br>To describe how the statistical models are validated |
| Chapter 6 | Results & Discussions | Evaluation Results | To statistically and schematically present the results of the performance evaluation of the proposed framework |
| | | Validation Results | To statistically and schematically present the results of proposed framework validation |
| | | Discussions | To synthesize the results of evaluation and validation |
| Chapter 7 | Conclusions | Aim and Objectives | To describe how aim and objectives of the study are attained |
| | | Contributions | To present contributions of the research |
| | | Significance | To highlight the significance of the work reported in this thesis |
| | | Publications | To present the list of publications produced from this thesis |
| | | Future Works | To identify the limitations and future work of the thesis |

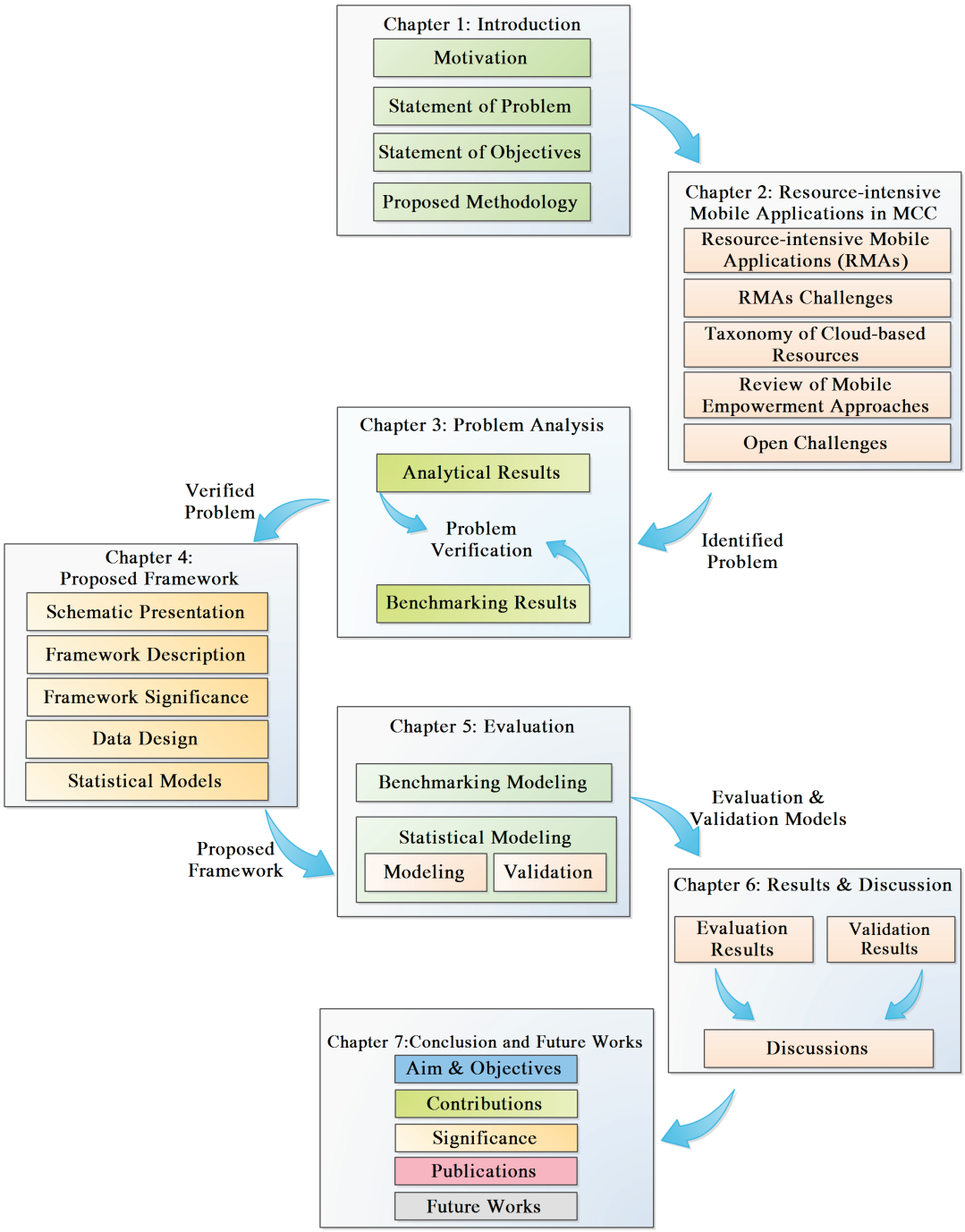



Figure 1.3: Schematic Representation of the Thesis Layout

marking experiments on real smartphone and mobile devices. Moreover, the statistical model that is derived for validation of the findings is described. We also describe the steps taken to validate the statistical models. Series of experiments to show platform-independence of our framework is described and the comparative study that is designed to demonstrate lightweight feature of our proposed framework is described. The chapter

is closed by describing the statistical methods used throughout this thesis to perform data analyses and syntheses.

Chapter 6 describes results on analysis of collected data to highlight the strength and weaknesses of our proposed framework. This chapter presents the results of our performance evaluation of the proposed model collected by analyzing two performance metrics, namely execution time and energy consumption of 30 workloads executed in local and remote modes using benchmarking analysis. The evaluation results are validated via statistical modeling and analysis. Finally, the synthesis of the benchmarking results and statistical modeling to demonstrate the validity of our proposed model are presented. The results of platform-independence experiments and comparative study are also presented in this chapter.

We conclude this thesis in chapter 7 by describing the efforts undertaken in this research to fulfill our aim. Also, we explain how the objectives of the research are fulfilled. The contributions of the thesis are presented and significance and strength of the proposed work are highlighted. Peer-reviewed international scholarly publications, including journal and conference articles are listed and limitations of the study are identified.

# CHAPTER 2

# RESOURCE-INTENSIVE MOBILE APPLICATIONS IN MOBILE CLOUD COMPUTING: A REVIEW

This chapter presents an overview of RMAs including their major requirements, and explores various challenges that mitigate efficient RMA execution and hinder successful RMA deployment and development. The most credible state-of-the-art research efforts aiming to empower mobile devices and applications are analyzed and synthesized to devise a taxonomy. The results of comparison of the reviewed efforts are also presented. Furthermore, we highlight several open issues as future research directions.

The remainder of this chapter is as follows. Section 2.1 introduces the RMAs and present existing challenges in efficient execution of RMAs. Four types of cloud-based computing resources are identified and taxonomized in section 2.3. Section 2.4 present review of the state-of-the-art mobile enhancement approaches based on varied cloud-based resources and present the devised taxonomy. Major open research challenges are presented in section 2.5 and the chapter is concluded in section 2.6.

## 2.1 Resource-intensive Mobile Application

RMAs are mobile applications that require intensive computational resources, particularly CPUs, RAMs, storage, and battery to successfully complete the expected computing operation. Image processing applications, enterprise systems, 3-D rendering applications, and video editing applications are examples of the RMAs. Functionalities and operations of RMAs are currently limited due to resource poverty of mobile devices. Execution of currently available RMAs immediately drains the mobile battery that significantly degrades the quality of user interaction. Hence, mobile resource augmentation becomes

necessary.

Mobile device is any non-stationary, battery-operating computing entity able to interact with end-user and execute transactions, store data, and communicate with the environment using wireless technologies and varied sensors. Smartphone, tablet, handheld/wearable computing devices, and vehicle mount computers are mobile device instances.

RMAs are comprised of varied combination of three major tasks, namely computation-intensive, data-intensive, and communication-intensive tasks. For instance, image processing RMAs employ mainly computational-intensive tasks such as arithmetic and logical tasks. In performing computation-intensive tasks, the mobile device requires extensive and long-lasting processing resources, particularly CPU and Graphical Processing Unit (GPU), main memory (i.e., RAM), and battery. Data-intensive tasks demand extensive storage resources and communication-intensive tasks need high performance wireless networking technologies and infrastructures. Communication-intensive tasks are interactive components of interactive applications that demand numerous call between client and server. Intensive communications between mobile and remote server remarkably prolongs application execution time and degrades application responsiveness and energy efficiency.

## 2.2 RMAs' Challenges

Mobile end-user requirements and expectations beside emerging heavy application development tools and technologies is insatiably increasing RMAs' resource requirements in mobile devices by demanding long-lasting, intensive computing resources. Mobile end-users demand extensive and accurate functionality, rich user interface, crisp responsiveness (response time of less than 150 ms (Tolia, Andersen, & Satyanarayanan, 2006)), context-awareness, offline usability, ubiquitous functionality and data access, device-independent

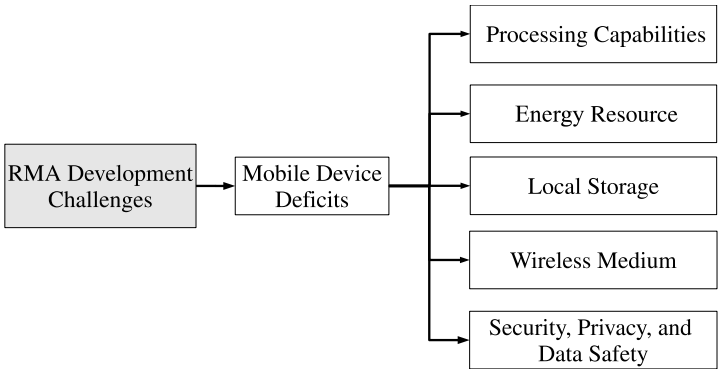



Figure 2.1: Taxonomy of RMA Development Challenges

functionality, and uninterruptible execution (Norman & Draper, 1986; Makris, Skoutas, & Skianis, 2013). However, in the absence of sufficient computing resources, RMA execution is inhibited and user experience is degraded. We extensively survey the literature and identify the vital challenges that hinder development and success of RMAs to devise a taxonomy. The devised taxonomy is illustrated in Figure 2.1 and described as follows.

1. **Limited Processing Capabilities:** Users constantly envision using smartphones with similar computing capabilities of desktop machines to perform heavy computing tasks while they are mobile. Such vision requires energy efficient, powerful processor and large memory. Though processing abilities of smartphones have always been increasing, user expectations (especially business users) are still far beyond processing capabilities of smartphones.

2. **Limited Power Source:** Energy is the only non-replenishable resource in smartphones that requires external resource to be renewed (Satyanarayanan, 2005). Smartphone manufacturers aim to attain device handiness, so bulk battery cannot be utilized. Moreover, battery capacity growth is about 5% annually (Robinson, 2009) since battery cells are excessively dense (Satyanarayanan, 2005). Sundry energy harvesting efforts (Flinn & Satyanarayanan, 1999; Starner, Kirsch, & Assefa, 1997; R.Avro, 2009) sought to replenish energy from renewable resources like human

movement, solar energy, and wireless radiation, but these resources are mostly intermittent and not available on-demand (Pickard & Abbott, 2012). Hence, restraint energy resources of smartphones remain a challenge to develop rich applications.

Recently, storage transactions and wireless communication are identified as the most energy-hungry tasks in smartphones. For instance, every 1 MB of data storage/retrieval consumes about 500 Mill watt of energy (Perrucci, Fitzek, & Widmer, 2011). Energy-aware algorithms and context-aware selection of communication medium from pool of heterogeneous technologies are effective ways to conserve mobile battery which are under investigation in next generation of wireless systems (Akyildiz, Jiang, & Mohanty, 2004; R. Y. Kim & Mohanty, 2010).

3. **Limited Local Storage:** Drastic increase in number of applications and amount of digital contents (Gantz et al., 2008) decelerate smartphones usability due to limited storage. While PCs are able to store huge amount of data inside the local hard disk, the smartphones are limited to few gigabytes of space which are mostly occupied by system files, user applications, and personal data. Therefore, frequent storing, updating, and deleting data, and uninstalling & reinstalling applications due to space limitation cause irksome impediments for mobile users and limit usability of smartphones.

   Additionally, delivering offline usability, which is one of the most important characteristics of RMAs, requires large local storage which smartphones lack. Storing partial content in device (Natchetoi, Kaufman, & Shapiro, 2008) aims to overcome this challenge. For example, instead of storing all emails locally, unread and unreplied emails are saved locally. Another feasible approach to augment local storage of smartphones is to utilize giant cloud storage similar to (Zheng, Xu, Huang,

& Wu, 2010; Wang, Wang, Ren, & Lou, 2009) proposals, but cloud computing still requires further advancement to be widely deployable in smartphones.

4. **Wireless Medium:** Wireless networks are intermittent and unreliable as compared to wired networks with high latency time, jitter, and non-uniform bandwidth that reduce quality of connectivity and prolong application responsiveness. To achieve crisp response, AJAX technology is leveraged to considerably hide latency time from end-user (Lawton, 2008); while server communication is performing in background, user can interact with the application. Consequently, transmission traffic and delay are reduced due to elimination of frequent screen refreshing (Deitel & Deitel, 2008). Cloudlet (Satyanarayanan et al., 2009) aims to achieve high quality of satisfaction and offer crisp response to multimedia and delay sensitive applications by performing remote computing-intensive tasks. Authors leverage virtualization technology, wireless LAN, and trusted resource-rich computing machine(s) in vicinity to shrink long WAN latency and jitter. However, it requires dramatic efforts to achieve crisp response of less than 150 ms (Tolia et al., 2006).

5. **Security, Privacy, and Data Safety Risks:** The dramatic increase in cyber crime and security threats in online transactions make security more challenging than ever (Cachin & Schunter, 2011). Security and privacy of personal data, financial records, user's online behaviors, and their location information are major concerns among mobile users while using smartphone applications (Prosper Mobile Insights, 2011). Information stored in smartphone is susceptible to security and safety breach due to high chance of robbery, physical damage, device failure, and loss. Security issues in the wireless medium, necessity of simple UI in mobile applications, and paying less attention to security principles by users increase security risks like exposing

user to fishing hazard (Whitten, 2004). For instance, hiding the bank's web link beneath an icon facilitates user to read and login quickly, but hidden bank address is susceptible to change to a fake link in absence of user perception (Bratus, Masone, & Smith, 2008). Moreover, features like GPS and accelerometer in smartphones, can potentially violate user security and privacy (Marquardt, Verma, Carter, & Traynor, 2011). This insecure platform makes smartphone less trustworthy ground and hence, impact on rapid spreading and development of RMAs in real scenarios (Marquardt et al., 2011; Bratus et al., 2008; Khan, Mat Kiah, Khan, & Madani, 2013).

Amalgam of these challenges has been encouraging researchers to alleviate the cloud computing technology (particularly, cloud-based resources) for mobile devices that has bred the state-of-the-art MCC toward augmenting computing capabilities of mobile devices for executing RMAs.

### 2.3 Taxonomy of Cloud-based Computing Resources

MCC realizes its vision by employing and integrating cloud-based resources with mobile augmentation solutions that is known as Cloud-based Mobile Augmentation (CMA). CMA is the-state-of-the-art mobile augmentation model that leverages cloud computing technologies and principles to increase, enhance, and optimize computing capabilities of mobile devices by executing RMA components in the resource-rich cloud-based resources.

The state-of-the-art research efforts (Cuervo et al., 2010; Satyanarayanan et al., 2009; Verbelen, Simoens, De Turck, & Dhoedt, 2012; Hung, Shih, Shieh, Lee, & Huang, 2011; Kosta, Aucinas, Hui, Mortier, & Zhang, 2012; Guo et al., 2011; Zhang, Kunjithapatham, Jeong, & Gibbs, 2011b; Chun et al., 2011; March et al., 2011; Badidi & Taleb, 2011; R.Kemp, Palmer, Kielmann, & Bal, 2010; Ma & Wang, 2012; Verbelen et al., 2012; Gu, March, & Lee, 2012; Xia et al., 2013) are aimed to realize user requirements and pref-

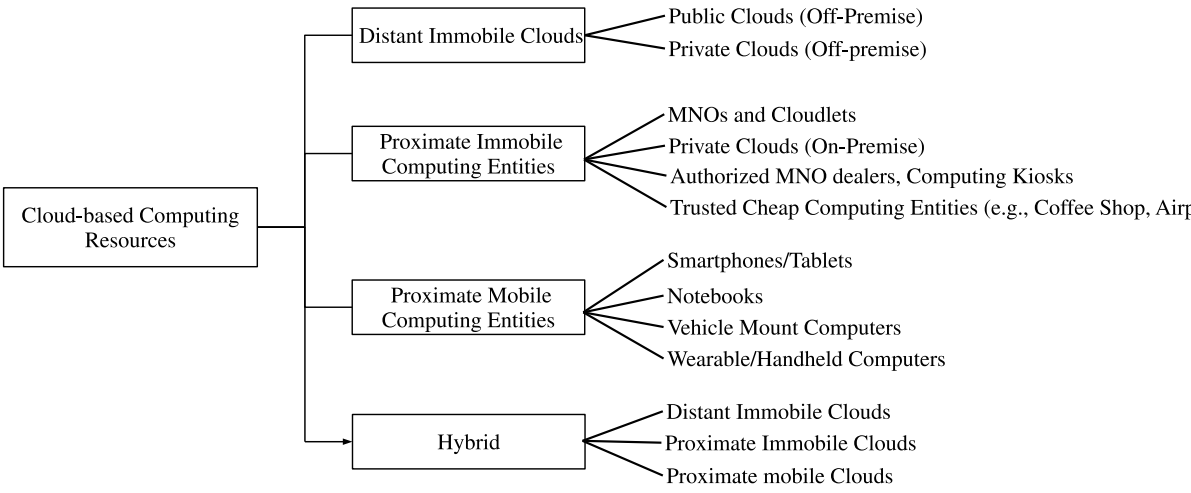



Figure 2.2: Taxonomy of Cloud-based Computing Resources

erences by exploiting varied types of cloud-based resources to enhance computing capabilities of resource-constraint smartphones. Based on the distance and mobility traits of such varied cloud-based computing resources, we classify them into four groups, namely distant immobile clouds, proximate immobile computing entities, proximate mobile computing entities, and hybrid that are taxonomized in Figure 2.2 and explained as follows.

Table 2.1 represents the comparison results of these cloud-based computing resources. While presenting description of each cloud-based type, we highlight their important features and present more exhaustive list of their features in the Table 2.1. This Table can be utilized as a guideline for appropriate selection of cloud-based infrastructures in future CMA researches.

### 2.3.1 Distant Immobile Clouds

Public and private clouds comprised of large number of stationary off-premise servers located in vendors or enterprises premises (i.e., not in the cloud consumer premise) are classified in this category. They are highly available, scalable, and elastic resources that

Table 2.1: The Comparison Results of Varied Cloud-Based Servers

| | Distant clouds | Proximate immobile computing entities | Proximate mobile computing entities | Hybrid |
|---|---|---|---|---|
| Architecture | Distributed | | | |
| Proximity | Low | Medium | High | Medium |
| Ownership | Service provider | Public | Individual | Hybrid |
| Environment | Vendor Premise | Business Center | Urban Area | Hybrid |
| Availability | High | Medium | Medium | High |
| Scalability | High | Medium | Medium | High |
| Sensing Capabilities | Medium | Low | High | High |
| Utilization Cost | Pay-As-You-Use | | | |
| Computing Heterogeneity | High | Medium | High | High |
| Computing Flexibility | High | Medium | High | High |
| Power Efficiency | High | Medium | Medium | High |
| Execution Performance | High | Medium | Medium | High |
| Security | Trusted | | | |
| Utilization Rate | High | | | |
| Execution Platform | VM | VM | Physical/VM | Physical/VM |
| Resource Intensity | High | Moderate | Moderate | Rich |
| Complexity | Low | Moderate | Moderate | High |
| Communication Technology | 3G/Wi-Fi | Wi-Fi | Wi-Fi | 3G/Wi-Fi |
| Communication Latency | High | Low | Low | Moderate |
| Execution Latency | Low | Medium | Medium | Low |
| Maintenance Complexity | Low | Medium | Medium | High |

are often located far from the mobile nodes accessible via the Internet. Although public cloud resources are likely more secure compared to the other types of resources due to complex security provisions and on-premise infrastructures (security, 2013; Kamara & Lauter, 2010; Wang et al., 2009; Mather, Kumaraswamy, & Latif, 2009), they are vulnerable to security attacks and breaches like Amazon EC2 crash (Cachin & Schunter, 2011) and Microsoft Azure security glitch (J. Clark, 2013). Accessing cloud resources, especially public clouds often carries the risk of communicating through the risky channel of Internet (Dikaiakos, Katsaros, Mehra, Pallis, & Vakali, 2009). However, giant clouds are endeavoring to maintain security –for more market share–and could establish high reputation-based trust by providing long-term services to the users.

Additionally, the performance and efficacy of these approaches are affected by long WAN latency due to the long distance between mobile client and stationary cloud data centers. One potential approach to shorten the distance between mobile device and cloud is to migrate the remote code and data to the computing resources near to the mobile

device via live migration of the VM from the cloud (C. Clark et al., 2005). However, live migration of VM is a non-trivial task that requires great deal of research and development, particularly in networking environment due to several issues such as large VM size, hard-to-predict user mobility pattern, and limited, intermittent wireless bandwidth.

Resource utilization is enhanced in clouds due to the virtualization technology deployment and emerging cloud resource scheduling algorithms (Cordeschi, Shojafar, & Baccarelli, 2013; Javanmardi et al., 2014). Several VMs can be executed on a single host to increase the utilization efficiency of the clouds, while each computation task runs on a single isolated VM loaded on a physical machine. However, VM security attacks such as VM hopping and VM escape (Owens, 2011) can violate the code and data security. VM hopping is a virtualization threat to exploit a VM as a client and attack other VM(s) on the same host. VM escape is the state of compromising the security of the hypervisor and control all the VMs.

### 2.3.2 Proximate Immobile Computing Entities

The second type of cloud-based computing resources involves stationary computers located in the public places near the mobile nodes, including on-premise private clouds that are built inside the premise of cloud service consumer. The number of computers in public places such as shopping malls, cinema halls, airports, and coffee shops is rapidly increasing. These machines are hardly performing tense computational tasks and are mostly playing music, showing advertisement, or performing lightweight applications. Moreover, they are connected to the power socket and wired Internet. Therefore, it is feasible to leverage such abundant resources in vicinity and perform extensive computation on behalf of resource-constraint mobile devices. It can also reduce latency and wireless network traffic while increases resource utilization toward green computing. Another group of proximate

immobile computers are Mobile Network Operators (MNO) and their authorized dealers scattered in urban and rural areas, private clouds, and public computing kiosk (Garriss et al., 2008) that can be exploited in smartphone augmentation.

However, protecting security and privacy of mobile user and computer owner hinder utilization of such nearby resources. Several shortcomings such as insufficient on-premise security infrastructure, lack of tight security mechanisms, and inefficient update and maintenance procedures inhibit utilizing such resources (except MNOs) for CMA approaches. Owners of these resources may attack mobile users and access their private data on the mobile devices or falsify offloading results. Also, malicious users may leverage these resources as an attacking point to violate mobile users' security and privacy. On the other hand, security and privacy of resource owners are also susceptible to violation. Owners of computer devices participating in resource sharing require robust mechanisms to protect and isolate the guest code and data from their host applications and data. Virtualization aims to realize such isolation mechanism, but issues such as VM hopping and VM escape require to be addressed before its successful adoption (Owens, 2011). Among all proximate immobile resources, MNOs may be considered unique in terms of security and privacy features. MNOs, in general, have been serving mobile users for long time and could establish high degree of trust among mobile users. It is feasible to assume that MNO's certified dealers also can inherit MNO's trust if central management and monitoring process is undertaken by MNOs.

### 2.3.3 Proximate Mobile Computing Entities

In this category of cloud-based infrastructures, various mobile devices, particularly smartphones, tablets, notebooks, wearable computers, and handheld computing devices play the role of servers based on cloud computing principles. The main benefit of utiliz-

ing nearby mobile resources is their proximity to the mobile clients. Also, hardware and platform heterogeneity between mobile servers and clients can be mitigated, because both sides are mainly ARM-based devices with mobile OSs. Moreover, contemporary smartphones are able to provide value added context- and social-aware services (Lane et al., 2010; Lukowicz, Pentland, & Ferscha, 2012) that contribute to the context-awareness of mobile applications. However, mobile devices' resources are limited and they are unable to perform intensive context-computing (Makris et al., 2013). Realizing distributed computing on cluster of nearby mobile devices requires several issues, particularly application architectures, resource scheduling, and mobility to be addressed.

Moreover, security and privacy of mobile devices as a service provider is a critical concern in CMA. Mobile devices are intrinsically susceptible to loss and robbery, and their constraint resources inhibit exploiting robust security mechanisms inside the device. Furthermore, with ever-increasing popularity of mobile Apps (i.e., mobile applications) in online App stores such as Google Play and Samsung APPs (Shen & Blau, 2012) number of mobile security threats are rising sharply and malware-contaminated Apps are becoming serious threats to the mobile users (MONACO, 2012). Several security threats have been identified in an experiment of Android mobile applications with the potential to violate the security of mobile users (Enck, Ongtang, & McDaniel, 2011). Risk of such contaminated codes can likely be transferred to the non-contaminated mobile devices by utilizing their computation resources and request for results of a remote computation. Hence, establishing trust between mobile devices and end-users becomes a challenging task.

### 2.3.4 Hybrid (Converged Proximate and Distant Computing Entities)

Hybrid infrastructures as depicted in Figure 2.3 are comprised of various proximate and distant computing nodes, either mobile or immobile. The main idea behind building

hybrid resources is to employ heterogeneous computing resources to create a balance between user requirements (mainly latency and computation power) and available options (Rahimi, Venkatasubramanian, Mehrotra, & Vasilakos, 2012). The latency sensitive codes are offloaded to the nearest computing device(s) whereas the most intensive and least latency sensitive tasks are migrated to the furthest resources. Perhaps, the utilization costs of nearby resources are more than the remote servers.

Beneficial characteristics of hybrid resources summarized in Table 2.1 advocates their usefulness in maximizing the augmentation benefits. However, deployment, management, and resource scheduling processes in dynamic mobile environment are non-trivial tasks. Developing an autonomic management system similar to CometCloud (H. Kim & Parashar, 2011) in cloud computing and MAPCloud (Rahimi et al., 2012) in MCC to automatically manage, optimize, and adapt hybrid infrastructures in the cloud-mobile applications can significantly improve the quality of hybrid CMA approaches.

Hybrid cloud infrastructures can deliver enhanced security and privacy features to the CMA approaches and increase the QoS because Hybrid resources consist of varied types of resources each of which featuring certain security and privacy aspect. Hybrid clouds are comprised of resources with varied security, privacy, and trust features (Takabi, Joshi, & Ahn, 2010; Ren, Wang, & Wang, 2012) which can be efficiently utilized by CMA and mobile users as a trade-off. For instance, security sensitive computations can perform a security-latency trade-off and execute computation inside a secure distant cloud (Satyanarayanan et al., 2009).

## 2.4 Taxonomy of the State-of-the-art Cloud-based Mobile Augmentation Approaches

Cloud-based Mobile Augmentation (CMA) is the-state-of-the-art mobile augmentation model that leverages cloud computing technologies and principles to increase, en-


Figure 2.3: The Hybrid Cloud Concept for MCC.

hance, and optimize computing capabilities of mobile devices by executing resource-intensive mobile application components in the resource-rich cloud-based resources. According to our resource classifications in section 2.3, we analyze and taxonomize the state-of-the-art CMA approaches into four models, namely distant fixed, proximate fixed, proximate mobile, and hybrid which are depicted in Figure 2.4. For each model, we describe few CMA efforts and tabulate the comparison results in Figure 2.5.

### 2.4.1 Distant Fixed

Majority of recent CMA approaches (Kemp, Palmer, Kielmann, & Bal, 2012; Guo et al., 2011; Zhang et al., 2011b; Chun et al., 2011; Verbelen et al., 2012; Di Francesco, 2012) leverage fixed cloud infrastructures in distance due to its straightforward approach. Utilizing stationary cloud eliminates several management complexities (e.g., resource discovery and scheduling for mobile cloud-based servers) and alleviates reliability and security concerns (Kristensen, 2010). Works in this class of CMA systems aim at reducing the complexity and overhead of utilizing distant cloud. For instance, in (Di Francesco, 2012)

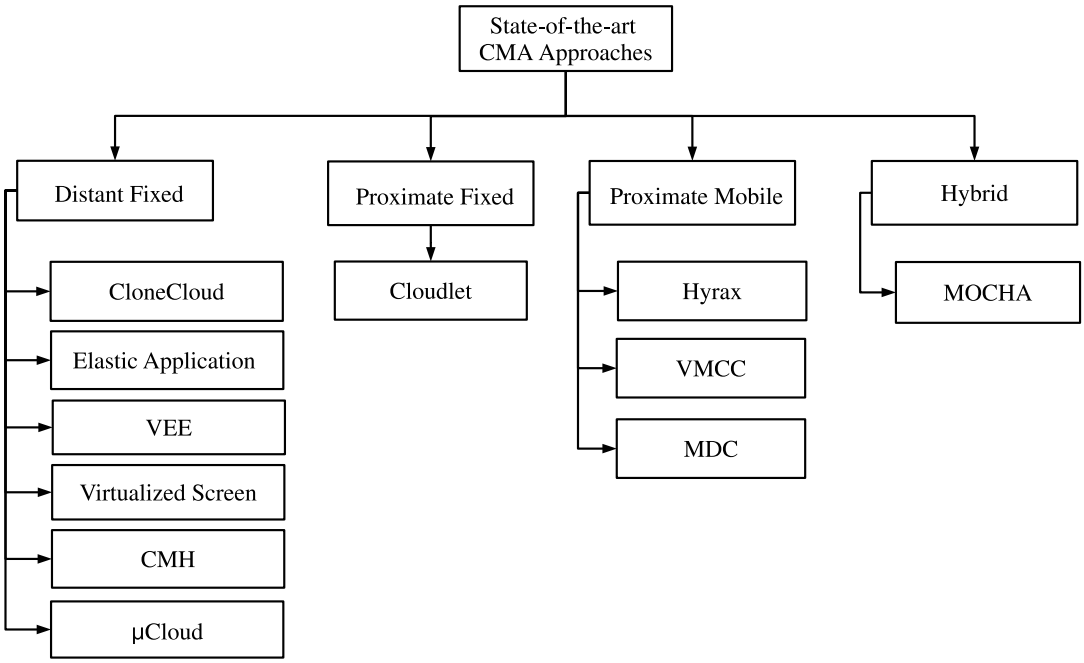



Figure 2.4: Taxonomy of State-of-the-art CMA Models.

authors propose an energy-efficient offline job scheduling model based on makespan minimization model to enhance energy efficiency of distant fixed CMA systems. Their main notion is to separate the data transmission from the job execution. During their work, authors provide several optimization solutions aiming to reduce the energy consumption of the device during the offloading process. However, for the sake of simplicity, the authors study the energy consumption of tasks in offline mode only which does not consider runtime dynamism of MCC. Exploiting cloud resources is feasible in several real scenarios such as live cloud streaming (Lawton, 2012), enterprise applications (e.g., Customer Relation Management (CRM) and enterprise resource planning (Hariharan, 2008)), and Social Networking. Cloud streaming mechanism is an example of utilizing distance fixed resources. In live cloud streaming approaches (Lawton, 2012), mobile device acts as a dump client able to interact with server using a browser or application Graphical User Interface (GUI). In live cloud streaming applications, entire processing take place in the cloud and results are streaming to the mobile devices. However, usability of cloud-streaming is hindered by latency, network bandwidth, portability, and network traffic cost. Functionality

of cloud-streaming applications absolutely depends on the network availability and the Internet. Transferring mobile-user input to the server is another critical factor that requires considerable attention under wireless Internet connection. Moreover, since majority of mobile network providers deploy 'pay-as-you-use' data plans, the large data traffic of cloud-streaming services imposes high communication cost on users. Yet congestion handling remains an open issue at peak hours. Entirely relying on cloud-streaming infrastructures and avoiding smartphones resources' utilization impact on application responsiveness and levy extravagant ownership, maintenance, power, and networking expenses to the cloud-streaming service providers, which is not a green computing approach.

In (Hariharan, 2008), researchers leverage cloud resources in developing a CRM application to enhance efficiency of sale representatives for a pharmaceutical company. The representative meets the physician in medical centers to promote drugs, present samples and promotions material, and he records all sale results and details through the mobile application. The huge database of the company is stored inside the cloud and the sale representative can request to process, get, or update data in database without storing data locally.

We describe some of the distant fixed CMA approaches that utilize distant fixed cloud resources for mobile augmentation as follows. The terms immobile, fixed, and stationary are interchangeably used with the same notion.

• *CloneCloud:* CloneCloud (Chun et al., 2011) is a cloud-based, fine-grained, thread-level, application partitioner and execution runtime that clones entire mobile platform into the cloud VM and runs the mobile application inside the VM without performing any change in the application code. The CloneCloud enables local execution of remaining mobile application when remote server is running the intensive components unless local execution tries to accessing the shared memory state. Cloud resources in this effort

simulate distributed execution of a monolithic application in a resourceful environment without engaging application developer into the distributed application programming domain. CloneCloud can significantly reduce the overall execution time using thread-level migration. When the local execution reaches the intensive component(s), the CloneCloud system offloads the component(s) to the cloud and continues local execution until the application fetches data from the migrated state. The local execution is paused until the results are returned and integrated to the local application.

However, the communication overhead of transferring the clone of mobile platform, application, and memory state and frequent synchronization of the shared data between the mobile and cloud can shrink the power of cloud. Such overhead becomes more intense in case of heavy, data- and communication-intensive, and tightly coupled mobile applications where an alternative execution of resource-intensive and lightweight components exists. Frequent code and data encapsulation and migration, and mobile-cloud data synchronization excessively increase the communication traffic and impact on execution time and energy efficiency of the offloading.

• *Elastic Application:* Elastic application model (Zhang et al., 2011b) is a CMA proposal that leverages distant fixed cloud data center for executing resource-intensive components of the mobile application. Authors in this model partition a mobile application into several small components, called weblet. Weblets are created with least dependency to each other to increase system robustness while decrease the communication overhead and latency. The weblet execution is dynamically configured to either perform locally or remotely, based on the weblet's resource intensity, execution environment quality, and offloading objectives.

The distinctive attribute of this proposal is that application execution can be distributed among more than one machine and cooperative results can be pushed back to

the device. To achieve such goal, multiple elasticity patterns namely replication, splitter, and aggregator are defined. In replication pattern, multiple replicas of a single interface are executed on multiple machines inside the cloud. Hence, failure in one replica will not compromise the system performance. In splitter pattern, the interface and implementation are separated so that several weblets with varied implementations can share a single interface. In aggregator, the results of multiple weblets are aggregated and pushed to the device for optimized accuracy and efficacy.

The authors endeavor to specify the execution configurations (specifying where to run the weblets) at runtime to match the requirements of the applications and users. To enhance the overall execution performance and enrich user experience, the system is able to run the weblets both locally and remotely. A weblet can be executed remotely in a low-end device while the same can be executed locally on a high-end device.

Elastic application model pays more attention to the user preferences by enabling different running modes of a single application (e.g., high speed, low cost, offline mode). Although weblet characteristics are mainly inherited from the well-known web services, determining weblets organization based on the functionality, resource requirements, and data dependency impose burden on programmers.

• *Virtual Execution Environment(VEE):* Hung et al. (Hung et al., 2011) propose a cloud-based execution framework to offload and execute the intensive Android mobile applications inside the distant cloud's virtual execution environment. The quality and accuracy of execution environment is highly influenced by the comprehensiveness and accuracy of emulated platform. This method uses a software agent in both mobile and cloud sides to facilitate the overall system management. The agent in mobile device initiates VM creation and clones the entire application (even native codes and UI components) and partial data/memory state from device to the cloud. Unlike CloneCloud, VEE aims to re-

duce latency by migrating the segment of data stack explicitly created and owned by the application to the VM instead of copying the entire memory; cloning the entire memory state, especially for heavy applications significantly increases latency and traffic.

During remote execution, the system frequently synchronizes the changes between device and cloud to keep both copies updated. In order to increase the quality and efficiency of remote execution in virtual environment and avoid data input loss at application suspension stages, the system stores input events (reading a file, capturing a face, storing a voice) exploiting a record/replay scheme and pseudo checkpoint methods. However, these methods engage application developers to separate the application state into two states, namely global and local, and to specify the global data structures. The global state contains the program domain and application flow, whereas the local state contains local data structures required by a method. Programmer usually needs to identify global state when the application is paused. Once the application is suspended, the global state will be loaded to avoid re-execution and the latest Android checkpoint is applied to the system to reflect all the changes made from the last checkpoint. However, all changes, especially user input might be lost from the last checkpoint. To record the changes after the last checkpoint, the record/replay mechanism is deployed by creating a pseudo checkpoint. To create a pseudo checkpoint, the application notifies the local agent to identify the input events and record required information. Upon the application resumption, the pseudo checkpoint is restored to restore the application to the state prior to the suspension.

In this effort, code security inside the cloud is enhanced by exploiting encryption and isolation approaches that protects offloaded code from cloud vendors eavesdropping. Using a probabilistic communication Quality of Service (QoS) technique, the authors aim to provide a communication-QoS trade-off. For instance, the control data (usually small volume) needs highest accuracy while video streaming data (often large volume) requires

less communication accuracy. Moreover, the authors are optimistic that offering secondary tasks such as automatic virus scanning, data backup, and file sharing in the virtual environment can enhance quality of user experience.

Although this approach aims to enhance the quality of application execution and augment computation capability of mobile clients and save energy, but responsiveness in interactive applications are likely low due to remote UI execution. Instead of migrating entire application to the cloud, it might be more beneficial to utilize some of the local mobile resources instead of treating mobile device as a dump client. Data passing between mobile device and cloud for interactive applications might degrade quality of experience, especially in low-bandwidth, intermittent networks.

• *Virtualized Screen:* Virtualized screen (Lu, Li, & Shen, 2011) is another example of CMA approaches that aims to move the screen rendering process to the cloud and deliver the rendered screen as an image to the mobile device. The authors aim to enrich the user experience by migrating the screen rendering tasks to the cloud with the assumption that majority of computation- and data-intensive processing take place in the cloud. Hence, abundant cloud resources' exploitation simplifies the CMA system architecture, prolongs mobile battery, and enhances the interaction and responsiveness of mobile applications toward rich user experience. Screen virtualization technique (running partial rendering in cloud and rest in mobile depending on the execution context) is envisioned to optimize user experience, especially for lightweight, high-fidelity, interactive mobile applications that entirely run on local resources. Their conceptual proposal aims to enhance visualization capability of mobile clients, mitigate the impact of hardware and platform heterogeneity, and facilitate porting mobile applications to various devices (e.g., smartphone, laptop, and IP TV) with different screens.

To reduce the mobile-cloud data transmission, a frame-based representation system is

exploited to forward the screen updates from the cloud to the mobile. Frame-based representation system captures and feeds the whole screen image to the transmission unit. This approach updates each frame based on the previous frame stored inside both the mobile and cloud. However, a rich interactive, responsive GUI needs live streaming of screen images which is impacted by communication latency. Although the authors describe optimized screen transmission approaches to reduce the traffic, the impact of computation and communication latency is not yet clear, as this is a preliminary proposal. Moreover, utilizing virtualized screen method for developing lightweight mobile-cloud application is a non-trivial task in the absence of its programming API.

• *Cloud-Mobile Hybrid (CMH) Application:* Unlike application offloading solutions, authors in this proposal (A.Manjunatha, A.Ranabahu, A.Sheth, & K.Thirunarayan, 2010) introduce a new approach of utilizing cloud resources for mobile users. In this effort, the authors propose a novel CMH application model, in which heavy components are developed for cloud-side execution, whereas lightweight or native codes are developed for mobile devices execution. CMH Applications execution does not need profiling, partitioning, and offloading processes and hence produce least computation overhead on mobile devices. Upon successful cloud-side execution, the results are returned back to the mobile for integrating to the native mobile components.

However, developing CMH applications is significantly complex due to the interoperability and vendor lock-in problems in clouds and fragmentation issue in mobiles. Cloud components designed for a specific cloud are not able to move to another cloud due to underlying heterogeneity among clouds. Similarly, mobile components developed for a particular platform cannot be ported to different platforms because of heterogeneity. Yet isolating development of mobile and cloud components creates further versioning and integration challenges.

To mitigate the complexity of CMH application developments and facilitate portability, the authors leverage Domain Specific Language (DSL)) (Ranabahu, Maximilien, Sheth, & Thirunarayan, 2011; van Deursen, Klint, & Visser, 2000). A DSL is a programming language with major focus on solving problem in specific domains. MATLAB is a well-known DSL-based tool for mathematicians. A parser takes a DSL script and converts codes into an in-memory object to be forwarded to various automatic component generators. The system needs different code generators for various mobile and cloud platforms. Once the mobile and cloud components are generated, the CMH application can be assembled for various mobile-cloud platforms. However, utilizing DSL-based techniques requires more generalization efforts to be beneficial in developing all types of CMH applications.

• µ*Cloud:* Similar to the CMH framework, µCloud (March et al., 2011) is a modular, mobile-cloud application framework that aims to facilitate mobile-cloud application generation, promote application portability, minimize the development complexity, and enhance offline usability in intensive mobile-cloud applications. Fulfilling separation of concerns vision, skilled programmers independently develop self-contained components which do not have any direct intercommunication with each other. Unskilled mobile users can mash-up (assembling available components to build complex application) these prefabricated components to generate a complex mobile-cloud application. Cloud vendors provide infrastructure and platform as cloud services to run prefabricated cloud components. The main idea in this proposal is to avoid local execution of the resource-intensive components. Hence, components are identified as cloud, mobile, and hybrid; mobile components are executable exclusively on mobile and cloud components are strictly developed for cloud server while hybrid components can either run locally or remotely. Hybrid components have either multiple implementations or a single implementation that need a mid-

dleware for execution. Each component has a triplet of identifier, input/output parameters, and configuration.

To alleviate offline usability issue, the authors leverage mobile-side queuing and cloud-side caching to maintain data in case of disconnection. Data will be transferred upon reconnection. Application is partitioned into components and organized as a directed graph. Nodes represent components and vertices indicate data/control flow. Application is divided into three fragments; in each fragment, a managing unit called orchestrator executes and maintains component's mash-up process. The output of each component is forwarded using the pass-by-value semantic as an input to the subsequent component.

Unlike elastic application model (Zhang et al., 2011b), the design and implementation of components in µCloud is statically performed in early development phase. Thus, any improvement in resource availability of mobile devices or environmental enhancement (like bandwidth growth) will not improve the overall execution of µCloud applications. Such inflexibility decreases the application execution performance and degrades the quality of user experience.

### 2.4.2 Proximate Fixed

Researchers have recently proposed CMA approaches in which nearby stationary computers are utilized. Utilizing nearby desktop computers initiates new generation of services to the end-user via mobile device. In (Satyanarayanan et al., 2009), the authors provide a real scenario in which Ron, a patient diagnosed with Alzheimer, receives cognitive assistance using an augmented-reality enabled wearable computer. The system consists of a lightweight wearable computer and a head-up display such as Google Glass[1] equipped with a camera to capture the environment and an earphone to send the feedback

[1] http://www.google.com/glass/start/

to the patient. The system captures the scene and sends the image to the nearby fix computers to interpret the scene in the image using the object or face recognition, voice synthesizer, and context-awareness algorithms. When Ron looks at a person for few seconds, the person's name and some clue information is whispered in Ron's ear to help greeting with the person. When he looks at his thirsty plant or hungry dog, the system reminds Ron to irrigate the plant and feed his dog. The nearby resources are core component of this system to provide low-latency real-time processing to the patient. In this part, we explain one of the most prominent proximate fixed efforts as follows.

• *Cloudlet:* Cloudlet (Satyanarayanan et al., 2009) is a proximate fixed cloud consists of one or several resource-rich, multi-core, Gigabit Ethernet connected computer aiming to augment neighboring mobile devices while minimizing security risks, offloading distance (one-hop migration from mobile to Cloudlet), and communication latency. Mobile device plays the role of a thin client while the intensive computation is entirely migrated via Wi-Fi to the nearby Cloudlet. Although Cloudlet utilizes proximate resources, the distant fixed cloud infrastructures are also accessible as backup in case of Cloudlet scarcity. The authors employ a decentralized, self-managed, widely-spread infrastructure built on hardware VM technology. Cloudlet is a VM-based offloading system that can significantly shrink the impact of hardware and OS heterogeneity between mobile and Cloudlet infrastructures.

To reduce the Cloudlet management and maintenance costs while increasing security and privacy of both Cloudlet host and mobile guest, a method called "transient Cloudlet customization" is deployed which uses hardware VM technology. It enables Cloudlet customization prior to the offloading and performs Cloudlet restoration as a post-offloading cleanup process to restore the host to its original software stake. The VM encapsulates the entire offloaded mobile environment (data state and code) and separates it from the host permanent software. Hence, feasibility of deploying Cloudlet in public places such

as coffee shops, airport lounge, and shopping malls increases.

Unlike CloneCloud and Virtual Execution Environment efforts that migrate the entire mobile OS clone to the cloud, Cloudlet assumes that the entire OS clone exists and is preloaded in the host and runs on an isolated VM. In mobile side, instead of creating the VM of the entire mobile application and its memory stack, the systems encapsulates a lightweight software interface of the intensive components called VM overlay.

The VM overall offloading performance is further enhanced by exploiting Dynamic VM Synthesis (DVMS) method since its performance solely depends on the mobile-Cloudlet bandwidth and cloudlet resources. DVMS assumes that the base VM is already available in the target Cloudlet and user can find the matching execution environment (VM base) among silo of nearby Cloudlets. Upon discovery and negotiation of the Cloudlet, the DVMS offloads the VM overlay to the infrastructure to execute launch VM (base + overlay). Henceforth, the offloaded code starts execution in the state it was paused. Upon completion of Cloudlet execution the VM residue is created and sent back to the mobile device. In the Cloudlet, the VM is discarded as a post-offloading cleanup process to restore the original Cloudlet state. In mobile side, the results will be integrated to the application and local execution will be resumed.

Despite the noticeable offloading improvements in the Cloudlet, its success highly depends on the existence of plethora of powerful Cloudlets containing popular mobile platforms' base VM. Encouraging individual owners to deploy such Cloudlets in the absence of monetary incentives is an issue that must be addressed before successful deployment in real scenarios. Although energy efficiency, security and privacy, and maintenance of Cloudlet are widely acceptable, further efforts are required to protect the overall CMA process. Moreover, few minutes offloading latency in Cloudlet is unusable to users (Tolia et al., 2006).

### 2.4.3 Proximate Mobile

Recently, several researchers (Marinelli, 2009; Mei, Taylor, Wang, Chandra, & Weissman, 2012; Huerta-Canepa & Lee, 2010; Guirguis, Ogden, Song, Thapa, & Gu, 2011) propose CMA approaches in which nearby mobile devices lend available resources to other mobile clients for execution of resource-intensive tasks in distributed manner. Utilizing such resources can enhance user experience in several real scenarios such as Optical Character Recognition (OCR) and natural language processing applications. The feasibility of utilizing nearby mobile devices is studied in (Huerta-Canepa & Lee, 2010) where Peter, a foreign tourist visiting a Korean exhibition finds interest in an exhibit, but cannot understand the Korean description. He can take a photo of the manuscript and translate it using the OCR application, but his device lacks enough computation resources. Although he can exploit the Internet web services to translate the text, the roaming cost is not affordable to him. Hence, he leverages a CMA solution by utilizing computation resources of nearby mobile devices to complete the task. Some of the CMA efforts whose remote resources are proximate mobile devices are explained as follows.

• *Hyrax:* Hyrax (Marinelli, 2009) is a CMA approach that exploits the resources from a cluster of immobile smartphones in vicinity to perform intense computations. Hyrax alleviates the frequent disconnections of mobile servers using fault tolerance mechanism of Hadoop. Similar to Cloudlet, due to resource limitations of smartphone servers, the accessibility to distant stationary clouds is also provisioned in case the nearby smartphone resources are not sufficient. However, Hyrax does not consider mobility of mobile clients and mobile servers. Hence, deployment of Hyrax in real scenarios may become less realistic due to immobilization of mobile nodes. Lack of incentive for mobile servers also hinders Hyrax success.

Hyrax is a MCC platform developed based on Hadoop (White, 2012) for Android

smartphones. In developing Hyrax, the MapReduce (Dean & Ghemawat, 2004) principles are applied utilizing Hadoop as an open source implementation of MapReduce. MapReduce is a scalable, fault-tolerant programming model developed to process huge dataset over a cluster of resources. Centralized server in Hyrax runs two client side processes of MapReduce, namely NameNode and JobTracker processes to coordinate the overall computation process on a cluster of smartphones. In smartphone side, two Hadoop processes, namely DataNode and TaskTracker are implemented as Android services to receive computation tasks from the JobTracker. Smartphones are able to communicate with the server and other smartphones via 802.11g technology.

Nevertheless, the cloud storage connectivity in Hyrax is missing. It demands several gigabytes of local storage to store data and computation. Hence, user cannot access distributed data over the Internet or Ethernet. The author utilizes the constant historical multimedia data to avoid file sharing. Hence, it is less beneficial for interactive and event-oriented applications whose data frequently changes over the execution and also data-intensive applications that require huge database. The overall overhead in Hyrax is high due to the intensity of Hadoop algorithm which runs locally on smartphones.

• *Virtual Mobile Cloud Computing (VMCC):* Researchers in (Huerta-Canepa & Lee, 2010) aim to augment computing capabilities of stable mobile devices by leveraging an ad-hoc cluster of nearby smartphones to perform intensive computing with minimum latency and network traffic while decreasing the impact of hardware and platform heterogeneity. During the first execution, required components (proxy creation and RPC support) are added to the application code to be used for offloading; the modified code will remain for future offloading. For every application, the system determines the number of required mobile servers, security and privacy requirements, and offloading overhead. The system continuously traces the number of total mobile servers and their geolocation to establish

a peer-to-peer communication among them. Upon decision making the application is partitioned into small codes and transferred to the nearby mobile nodes for execution. The results will be reintegrated back upon completion.

However, several issues encumber VMCC's success. Firstly, this solution, similar to Hyrax, is not suitable for a moving smartphone since the authors explicitly disregard mobility trait of mobile clients. Secondly, application code should be available for manipulation which is not always feasible. Thirdly, every computing job is sent to exactly one mobile node; so, the offloading time and overhead will be increased when the serving node leaves the cluster. Fourthly, the offloading initiation might take long since the offloading's overall performance highly depends on the number of available nearby nodes; insufficient number of mobile nodes defers offloading. Finally, in the absence of monetary incentive for mobile nodes the likelihood of resource sharing among resource-constraint mobile devices is low.

• *Mobile Device Cloud (MDC):* MDC is a computational offloading approach that builds a remote execution environment in convergence of a set of mobile devices belong to a particular individual or a corporate entity. MDC assumes highly collaborative mobile devices (e.g., smartphone, tablet, and laptop) where mobile devices collaboratively share their batteries so that all mobile devices finish battery at the same time. In this approach, the aim is to prolong the battery lifespan of all mobile devices to a certain extent so that no individual device dies earlier than others. Since there is no chance for strangers to participate in this network security is trustable if wireless network is secure.

However, adoption of this solution is limited to an individual or single company who wish to prolong life time of all his mobile devices using accumulation of batteries. This cloud of mobile devices remains alive until the first mobile device dies. Moreover, the energy requirement of every task in this environment should be known which is not trivial

task considering high heterogeneity in MCC. In this model there is no reconnection chance if the wireless connectivity dismiss which is very common in mobile environments. Another weakness in this work is that strangers cannot share their battery resources to each other, in exchange of either money or credit.

### 2.4.4 Hybrid

Hybrid CMA efforts are budding (Soyata, Muraleedharan, Funai, Kwon, & Heinzelman, 2012; Rahimi et al., 2012) to optimize the overall augmentation performance and researchers are endeavoring to seamlessly integrate various types of resources to deliver a smooth computing experience to mobile end-users. For instance, mCloud (Bahl, Han, Li, & Satyanarayanan, 2012) is an imminent proposal to integrate proximate immobile and distant stationary computing resources. Authors are aiming to enable mobile-users to perform resource-intensive computation using hybrid resources (integrated cloudlet-cloud infrastructures). Hybrid solutions aim to provide higher QoS and richer interaction experience to the mobile end-users of real scenarios explained in previous parts. For instance, in the foreign tourist example, the image can be sent to the nearby mobile device of a non-native local resident for processing. When the processing fails due to lack of enough resources, the picture can be forwarded to the cloud without Peter pays high cost of international roaming (Peter may pay local charge).

We review some of the available hybrid CMAs as follows.

• *MOCHA:* In MOCHA (Soyata et al., 2012) authors propose a mobile-cloudlet-cloud architecture for face recognition application using mobile camera and hybrid infrastructures of nearby Cloudlet and distant immobile cloud. Cloudlet is a specific, cheap cluster of computing entities like GPU capable of massively processing data and transactions in parallel. Cloudlets are able to be accessed via heterogeneous communication technolo-

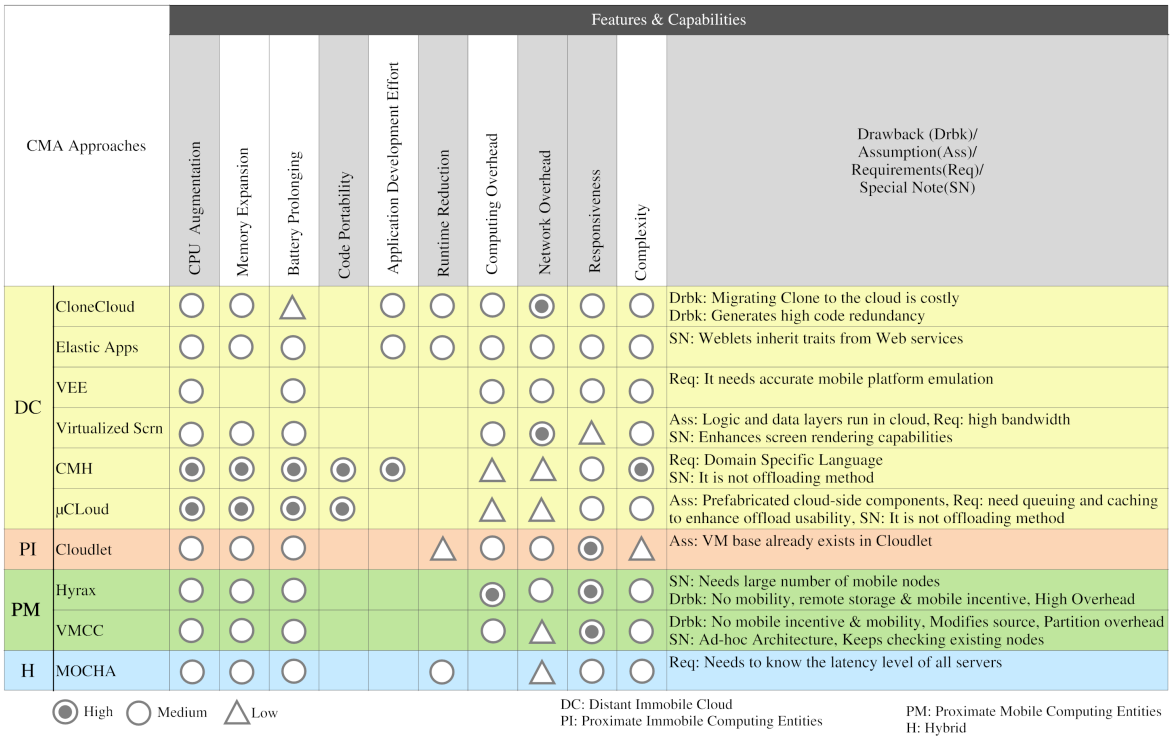



Figure 2.5: Comparison of CMA Approaches.

gies such as Wi-Fi, Bluetooth, and cellular. The mobile users often access processing resources via Cloudlet rather than directly connecting to the cloud —unless accessing cloud resources bears lower latency.

Cloudlet receives the smartphones' intensive computation tasks and partitions them for distribution between itself and distant immobile clouds to enhance QoS (Satyanarayanan et al., 2009). MOCHA leverages two partitioning algorithms: fixed and greedy. In the fixed algorithm, the task is equally partitioned and distributed among all available computing devices (including Cloudlet and cloud servers), whereas in greedy algorithm, the task is partitioned and distributed among computing devices based on their response times; the first partition is sent to the quickest device while the last partition is sent to the slowest device. The response time of the task partitioned using greedy approach is significantly better than fixed, especially when Cloudlet server is utilized in augmentation process and large number of clouds with heterogeneous response time exist.

However, smartphones in MOCHA require prior knowledge of the communication

and computation latency of all available computing entities (Cloudlet and all available distant fixed clouds) which is a resource-hungry and time-consuming task.

## 2.5 Open Research Challenges

In this section, we highlight some of the most important challenges in deploying and utilizing CMA approaches as the future research directions.

### 2.5.1 Reference Architecture for Mobile Augmentation Frameworks

Recently, researchers leverage dissimilar structures and techniques in utilizing cloud resources to augment computation capabilities of mobile devices. Also, different efforts focus on varied types of mobile applications, particularly multimedia, intense games, image processing, and workflow processing applications. Such diffusion scatters development approaches and increase adaptability challenges of existing frameworks. In the absence of a reference architecture and unified augmentation solution, various approaches need to be integrated to all mobile OSs to serve multi-dimensional needs of various mobile users which is a non-trivial task. The reference architecture is expected to be generic enough to be deployed in family of CMA approaches.

### 2.5.2 Long WAN Latency

Latency adversely impacts on the energy efficiency and interactive response of cloud-mobile applications by consuming excessive mobile resources and raising transmission delays. In wireless communication, distance from the access point (near or far) and variation in bandwidth and speed of various wireless technologies affect the energy efficiency and usability of mobile devices. Moreover, leveraging wireless Internet networks to offload RMAs to distant cloud resources creates a bottleneck. Consequently, the long WAN latency is increased while the quality of user experience is decreased. To reduce interaction latency, proximate resources can be near to optimal solutions if underlying technolo-

gies and architectures are lightweight. Proposals such as Cloudlet, Hyrax, and VMCC proposed to create a proximate cloud to access nearby remote resources, but further advancement is necessary to mitigate the operation overhead. Service oriented architecture and HTTP client-server communication using lightweight architectural style is potentially significant in alleviating the long WAN latency.

### 2.5.3 Lightweight CMA

Developing lightweight mobile computing augmentation approaches to increase quality of mobile user experience and to develop CMA system independent of any particular situation is a significant challenge in mobile cloud environment. Offloading bulk data in limited wireless bandwidth, and VM initiation, migration, and management in a secure and confidential manner are particular tasks in CMA system that noticeably increase overall execution time, intensify the augmentation latency, and decrease the quality of mobile user experience. CMA approaches are generally hosted and executed inside the smartphone conserving their local resources and so, need to avoid resource hungry tasks.

A feasible approach to decrease the volume of digital contents —in limited bandwidth networks —is to utilize effective, efficient data compression methods. Available compression techniques are unlikely efficient considering structure of current multimedia files. Moreover, approaches like Paravirtualiztion (Youseff, Wolski, Gorda, & Krintz, 2006) as a lightweight virtualization technique can reduce the overhead by partially emulating the OS and hardware. Paravirtualization approaches virtualize only parts of the hardware required for computing. Thus, the mobile-side VM creation overhead is diminished and the impact of VM migration on network is reduced. Therefore, realizing lightweight CMA approaches demands lightweight computation and communication techniques (particularly in virtualization, data compression, and encryption methods) to reduce intra-system cor-

respondence, data volume, and I/O tasks.

### 2.5.4 Computing and Temporal Cost of Mobile Distributed Execution

Noticeable computation and communication cost of migrating tasks from the mobile device to the remote servers and receiving the results is another challenge of CMA approaches in MCC, which is intensified by mobility and wireless communication constraints. Although researchers (Marinelli, 2009) endeavor to reduce the distance of mobile devices and service providers by leveraging nearby mobile/fixed computing devices, several mechanisms, particularly resource discovery and allocation, service consumer and provider mobility management, and distributed runtime are required to realize the CMA's vision. Accurate allocation of resources to the mobile computation tasks demands comprehensive knowledge about structure and performance features of available service providers and resource requirements of mobile computation tasks. Thus, QoS-aware scheduling efforts such as (Rahimi et al., 2012) are necessary to enhance the CMA usability.

## 2.6 Conclusions

In this chapter, an overview of RMAs is presented, current challenges that hinder their success are described, and most credible augmentation efforts in MCC are reviewed. We also identified several research problems of the domain. Our investigation results unveil that enhancing computing capabilities of mobile devices is feasible via four types of cloud-based resources, including distant and proximate resources. However, utilizing these resources to enhance computing power of mobile devices is not a straightforward panacea and originates several inefficiencies. One of the most significant inefficiencies in using distant cloud datacenters as remote resources is the associated long WAN latency. We found that although distant clouds feature high availability, resource elasticity, and scalability, utilizing resources in distance is associated with long WAN latency that sig-

nificantly impacts on prolonging application execution time, decreasing energy efficiency, and degrading user experience. Considering the significance of responsiveness and energy efficiency in adoption and success of RMAs in critical domains, particularly healthcare, education, and tele-operation, alleviating the impact of long WAN latency is a significant challenging problem of the domain that is considered to be addressed in this research.

# CHAPTER 3

# PERFORMANCE ANALYSIS OF RESOURCE-INTENSIVE MOBILE APPLICATIONS IN MOBILE CLOUD COMPUTING

In this chapter, we aim to analyze the performance of executing RMAs in remote cloud-based resources to investigate the impact of long WAN latency stemmed from long mobile-cloud distance. Using analytical analysis mathematical equations are derived to identify the contributing time-consuming components in execution of RMAs that demonstrate the significant of WAN latency when utilizing distant immobile resources. We formulate the round-trip latency of transmitting the mobile request to the cloud resources and receiving of the cloud response. The initial findings are verified via series of benchmarking experiments on real MCC environment. The results unveil that from two cloud VMs with identical computing capability, utilizing the proximate cloud significantly impacts on the application execution time and energy efficiency.

The remainder of this chapter is as follows. Section 3.1 presents our analytical analysis of WAN latency when using distant clouds for empowering mobile devices. We describe our benchmarking experiment in section 3.2 and the chapter is concluded in section 3.3.

## 3.1 Analytical WAN Latency Analysis

In this section, we analytically investigate the metrics contributing on WAN latency in MCC, especially when Distant Immobile Clouds (DIC) resources are utilized. We also investigate the impact of long WAN latency on application execution time and energy efficiency. We formulate execution time and energy consumption of RMAs that employ Distant Immobile Cloud (DIC) resources for execution to study the impacts of mobile-

cloud distance and number of intermediate hops on execution of RMAs on smartphones.

We further investigate the latency change when the packet size varies in the CMA system. During current system modeling, we consider math utility applications, namely prime and matrix multiplication that perform all the computations as server component and there is no native code to be executed. Data is migrated from the client to the server and results are sent back and displayed on the screen.

DIC is one of the most prevalent types of cloud resources used in CMA. DICs are providing high availability, elasticity, and scalability, but feature communication latency and lack mobility. DICs are located at cloud vendor premises far from the mobile users. Hence, when accessing their services, data and code travel long distance in network (wireless and wired) to reach the DICs and return to the mobile users, which is deemed to originate significant communication delay. When the distance between service provider and consumer increases, the number of intermediate hops that relay data packets between these two hikes. Considering the noticeable temporal overhead of intermediate hops to process packets (perform unmarshaling, decompressing, decrypting, processing, manipulating, encrypting, compressing, and marshaling), growth in number of intermediate hops significantly impacts on the communication latency and degrades efficiency of CMA systems. Thus, it is vital to analyze the impacts of distance and number of hops between DIC resources and mobile service consumers on overall execution performance of RMAs when execution is performed in the DICs.

### 3.1.1 Execution Time Analysis

The overall execution time in mobile-cloud applications is the sum of mobile computation, round trip delay, and cloud computation time represented in Equation (3.1). The mobile computation is the time to perform native computations. The round trip latency is

the delay of sending request and receiving response. The cloud computation delay is the time that request is processed in server side and results are produced. While we can shrink the volume of native computation and demand rich computing resources in the cloud, decreasing the round trip delay sounds trivial due to very large number of influential factors which will be briefly discusses as follows. The overall execution time is represented by $T_{total}$ as following:

$$T_{total} = T_{mobile} + T_{RT} + T_{cloud} \tag{3.1}$$

Where $T_{mobile}$ is the time to perform local computation that is negligible in our model, $T_{RT}$ is the round trip delay, and $T_{cloud}$ is the computing latency of the cloud. While computing latency depends on the computational capabilities (CPU clock speed, RAM volume, I/O performance, and environmental context like heat) of the computing server, round trip delay comprises of four delays, including transmission, propagations, processing, and queuing delay (Ramaswamy, Ning, & T.Wolf, 2004).

The following equation needs to be multiplied by 2 because in computation augmentation there are two sets of communications; once when the request is sent from the mobile device to the remote cloud server and another time when the response is sent from the cloud to the mobile device. Thus, we will have

$$T_{RT} = 2 \times (T_{prog} + T_{tran} + T_{proc} + T_Q) \tag{3.2}$$

where $T_{prog}$ or propagation delay is the time to transmit the packet through the medium from the client to the server, $T_{tran}$ or transmission delay is the time to send a packet onto the communication medium (wire/wireless). $T_{proc}$ or processing delay is the time the intermediate nodes take to handle the packet on the network, and $T_Q$ or queuing delay is the time packet is queued before transmission starts. Due to increasing complexity of data pro-

cessing during propagation stage, processing delay increased significantly. Although the processing power of intermediate network nodes is enriched, still the delay is noticeable. We discuss each type of delay as follows:

- **Propagation delay:** is directly related to the distance between two network nodes and is the time taken for the packet to travel from sender to the receiver. Hence, the propagation delay is sum of delay between each two nodes throughout the path between mobile and cloud. If we consider identical delay between each pair of nodes, it is essential to multiply the propagation delay into the number of hops. Moreover, HTTP protocol of TCP/IP needs establishment of connection prior the request and response transmission. We need to consider delay to establish and terminate the connection, and receive the acknowledgment. If $H$ represents the number of hops, we will have:

$$T_{prog} = 4 \times H \times \frac{D}{S} \tag{3.3}$$

  where D is the distance between each pair of nodes and S is the velocity factor or propagation speed of the signal in the medium which is considered to be the light speed in vacuum, i.e., $3 \times 10^8$.

- **Transmission delay:** $T_{tran}$ depends on the amount of data is transferred over the network bandwidth $\beta$. For each packet with *packet size* bit, the total transmission delay of P packets will be

$$T_{tran} = P \times (\frac{packet\ size}{\beta}) \tag{3.4}$$

- **Queuing delay:** is the amount of time taken in $H-1$ hops for transmission of

packet over the network with transmission delay of $T$. So, we will have

$$T_Q = (H-1) \times (\frac{packet\ size}{\beta}) \tag{3.5}$$

- **Processing delay:** When the client-server distance increases, there is high chance of increase in the number of intermediate networking hops. In every hop there is a processing delay of $T_{proc(i)}$ that are assumed to be identical. Thus, for a client-server distance consists of $H$ hops we have:

$$T_{proc} = 4 \times (H-1) \times T_{proc} \tag{3.6}$$

By substituting the equations 3, 4, and 5 in Equation (3.2), we will have

$$T_{RT} = 2 \times ((4 \times H \times \frac{D}{S}) + (P \times (\frac{packet\ size}{\beta})) + ((H-1) \times (\frac{packet\ size}{\beta})) + (4 \times (H-1) \times T_{proc})) \tag{3.7}$$

From the above equations, it is clear that the round trip latency highly depends on the number of hops between mobile and cloud, rather than the physical distance. Direct impact of distance on roundtrip latency is negligible considering the high transmission speed of current networks.

We demonstrate the latencies change by factors such as distance and number of hops using the following examples. Consider the following common characteristics for our examples.

- transmission speed of all links is $56 kbit/s$, equals to $56000 b/s$
- processing delay for all hops is $5 \mu s$, equals to 0.000005 seconds
- packets size is 1000 bytes long, equals to 8000 bits

- propagation speed is $3 \times 10^8$

In the first example, the distance between mobile and cloud is about 320 KM with 14 hops and in the second example, distance is 6500 KM and 24 hops. The number of packets for both examples is 20. Hence, we will have

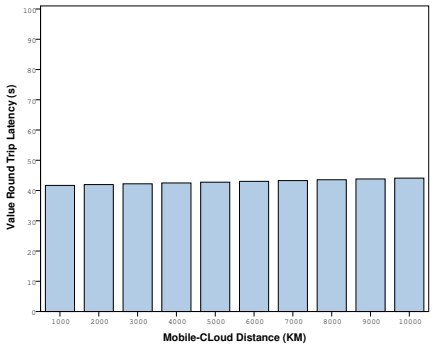
$$T_{RT(1)} = 2 \times ((4 \times 14 \times \frac{320 * 1000}{300000000}) + (20 \times \frac{8000}{56000}) + (13 \times \frac{8000}{56000}) + (4 \times 13 \times 0.000005)) = 9.54 \tag{3.8}$$

For the second example, we will have

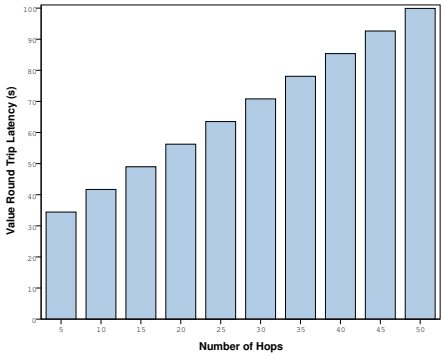
$$T_{RT(2)} = 2 \times ((4 \times 24 \times \frac{6500 * 1000}{300000000}) + (20 \times \frac{8000}{56000}) + (23 \times \frac{8000}{56000}) + (4 \times 23 \times 0.000005)) = 16.44 \tag{3.9}$$

The difference between the round trip latency of these two examples is as much as 73% which is remarkable difference. Equation (3.7) depicts that the main influential factor is the number of hops between the mobile and cloud computer (which is very closely related to the mobile-cloud distance) and ration of packet size to bandwidth. However, if the number of hops is fixed, increase in distance does not change the results much. To further infer results from our time model, we extend our analysis using SPSS. The results of our analysis are plotted in Figures 3.1.

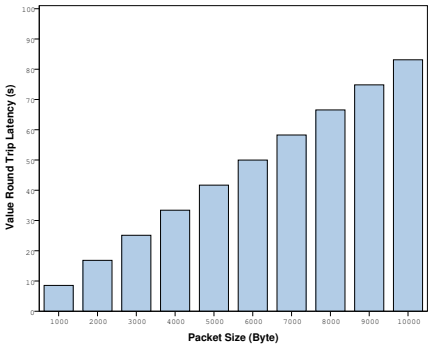
Figure 3.1(a) depicts round trip latency when mobile-cloud distance is increasing from 1000 to 10000 KMs with step of 1000 KMs. The number of hops is 10 and the packet size is 5000 bytes. As illustrated, there is almost no difference when the distance increases without raise in number of hops, because the transmission speed of today's network is remarkably high and increase in distance does not affect much on the latency. Figure 3.1(b) depicts round trip latency when the number of hops between mobile and




(a) Increase in Mobile-Cloud Distance




(b) Increase in Hops




(c) Increase in Packet Size

Figure 3.1: Round Trip Latency Analysis

cloud is increasing from 5 to 50. The distance between mobile and cloud is 1000 KMs and the packet size is 5000 bytes. As results indicate, number of hops significantly impacts on round trip latency. This delay is due to the processing overhead that exists when forwarding data packets at any hop. Figure 3.1(c) depicts round trip latency when data packet size changes from 1000 to 100000 bytes. The number of hops is 10 and the mobile-cloud distance is 1000 KMs. The Figure testifies the impact of packet size on the latency. Although the packet size itself has direct impacts on the latency, increase in packet size originates excess overhead at each hop to perform pre- and post-forwarding processes. Certainly, when the packet size increases, the overhead of excess header size, Cyclic Redundancy Check (CRC) and validation, compression, and encryption increase the processing time at each hop. Further analysis is undertaken using Matlab to better highlight the impacts of hop number and packet size which is illustrated in Figure 3.2. For sure, number of hops

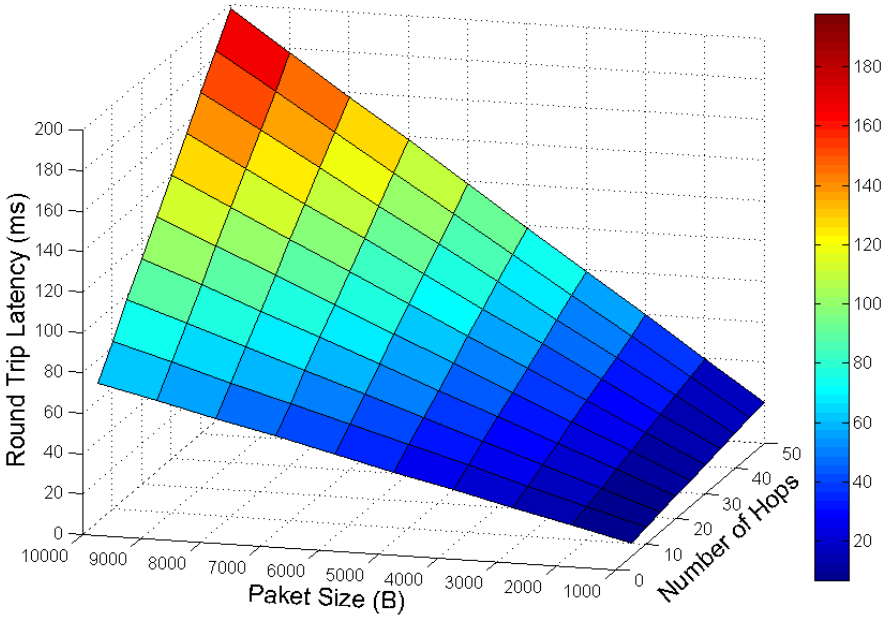



Figure 3.2: Results of Studying Correlations of Hop Number and Packet Size on Round-trip Latency for Cloud-based RMAs.

and packet size directly impact on round-trip latency for cloud-based RMAs. 3-D view of the analysis unveils that round-trip latency proportionately grows when number of hops and packet size rise. Hence, the number of mobile-cloud hops and data packet size are the most influential factors on round-trip latency.

### 3.1.2 Mobile Energy Consumption Analysis

The overall energy consumption of a typical client/server RESTful applications (i.e., $E_{total}$) is the sum of energy consumed to perform client-side computation ($E_{client}$), transmit request (including data) to the server ($E_{trans}$), wait to receive the response ($E_{wait}$), and reintegrate and synchronize the results with the client code which is formulated in Equation (3.10).

$$E_{total} = E_{client} + E_{trans} + E_{wait} + E_{Sync} \tag{3.10}$$

The energy consumption of client side can be divided into the following items:

$$E_{client} = E_{native} + E_{pre} \tag{3.11}$$

where $E_{native}$ is the energy consumed to run the native code which is negligible in our evaluation and $E_{pre}$ is the energy consumed to prepare the request for transmission which consists of tasks such as encryption and compression of data. To enhance energy efficiency of the mobile device, we have mitigated the amount of native computations almost to zero so that the overhead of transmitting the request does not exceed the computation overhead. The $E_{trans}$ is the amount of energy used to transmit the data to the cloud and receive the response. If $\gamma$ represents the energy consumption of transmitting a bit from client to server over the network, we will have:

$$E_{trans} = \gamma \times Content \tag{3.12}$$

The wait energy $E_{wait}$ is a function of computing latency that has direct correlation with the overall execution time of the application. Hence, we assume for each time unit (ms), there is $\alpha$ energy unit (mJ) consumed. So, we will have

$$E_{wait} = \alpha \times (T_{RT} + T_{cloud}) \tag{3.13}$$

By replacing round trip latency to the above equation, we will have

$$E_{wait} = 2\alpha \times (T_{prog} + T_{tran} + T_{proc} + T_Q) + \alpha \times T_{cloud} \tag{3.14}$$

If we replace right side of $E_{client}, E_{trans}, E_{wait}, and E_{sync}$ in Equation 3.10, we will have

$$E_{total} = E_{native} + E_{pre} + (\gamma \times Content) + E_{Sync} + 2\alpha(T_{prog} + T_{tran} + T_{proc} + T_Q) + \alpha T_{cloud} \tag{3.15}$$

In our applications which are developed using RESTful architectural design (Fielding, 2000), the native code and transmission overhead is very low since the code stored in the cloud and is not migrating in majority of cases (unless the server-side code is not accessible) and hence, it solidly depends on the amount of data being sent. Such lightweight transmission shrinks the value of $E_{native}$ and $E_{pre}$ since encryption or compressing small amount of data are not resource-intensive tasks. Also $E_{trans}$ highly depends on the amount of data to and from the mobile device. On the other hand, $\alpha T_{cloud}$ is very small considering the power of today's giant clouds. Therefore, as Equation (3.15) shows, the energy consumption of the mobile device is highly depends on the amount of data traffic and communication latency of the mobile application when using the cloud. From the time analysis, we know that the number of hops significantly increase the round trip latency. Hence, the same can degrade energy efficiency of mobile devices. Therefore, we conclude that the less number of intermediary nodes and amount of data transmission, the better would be energy efficiency of mobile-cloud applications. We validate results of our round trip latency model and mobile energy via benchmarking on real device that are presented as follows.

## 3.2 Empirical Experimentation

In order to validate our initial findings in previous section, we perform a series of benchmarking experiments. In these experiments, we investigate the impacts of CMA on resource conservation of mobile devices and analyze the effects of mobile-cloud distance and number of intermediate hops on augmentation performance using two series of benchmarking experiments. We employ Resource-Oriented Architecture (ROA) design principle in this work to omit the overheads of identifying, partitioning, and offloading platform-dependent application components from the mobile device to the DICs that is

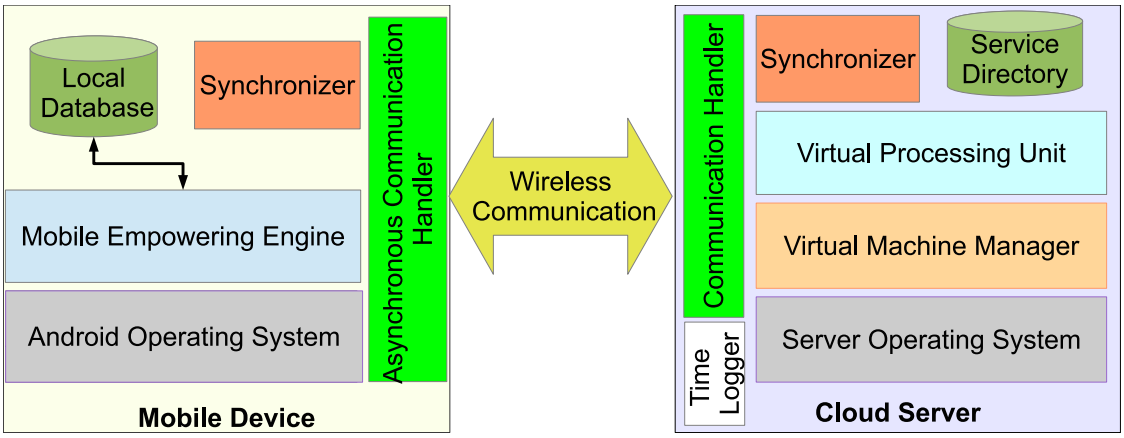



Figure 3.3: Graphical Representation of the Experimental Model

helping us to concentrate on studying the distance and number of hops in CMA process. We analyze and synthesize time and energy overheads of two RMAs in three execution strategies of native, proximate cloud, and distant cloud. Experimental model including mobile device specification, cloud servers, and communication tools are described. We explain our benchmark, data collection method, and result logging processes.

### 3.2.1 Model

Our benchmarking model is illustrated in Figure 3.3 followed by describing operations of each component in the model and implementation of the model.

A. **Mobile Device Components:** The mobile client device is a smartphone featuring a dual-core application processor with 1.2 GHz Exynos Cortex-A9 CPU and 1GB RAM. Above the hardware layer, the Android operating system 4.0.3 is located. The major components run on the mobile device are described as below.

**Mobile Empowering Engine:** Mobile Empowering Engine (ME2) is responsible to identify the binding method of the desired cloud servers by inquiring the local database. The local database contains the IP addresses of the available remote servers. At runtime, when the execution flow of

the RMA reaches the intensive component, the ME2 identifies the Internet Protocol (IP) address(es) of the server(s) to be called for remote execution. The ME2 embeds the data, IP address, and binding method into a request and forwards it to the asynchronous communication handler, so that the request can be forwarded to the server. Once the response is delivered from the server by the Synchronizer components, it reintegrates the response in to the system and presents it to the user on the device screen.

**Asynchronous Communication Handler:** Asynchronous Communication Handler (ACH) leverages the wireless communication interface in the mobile client device that receives the request from ME2 and asynchronously forwards the request to the server. Upon completion, the ACH receives the response from the server and forwards the results to the Synchronizer for the reintegration of the results to the system.

**Synchronizer:** The Synchronizer is responsible to maintain integration between the memory state of the client and the server. It receives the response from the server and reintegrates it with the native application so the ME2 can present the results to the user.

B. **Cloud Server Component:** In this experiment, we employ two cloud servers located in different geographical locations, namely Singapore (represents the proximate cloud) and Sydney (represents distant cloud) regions. Servers are featuring a t1.micro VM instance of Amazon EC2. Each VM instance features up to two EC2 compute units processor (about 1.0-2.4 GHz Opteron or Xeon), 613 MB RAM, and low performance I/O interface running Microsoft Windows Server 2008 Base 64-bit OS. Figure 3.4 depicts the geo-

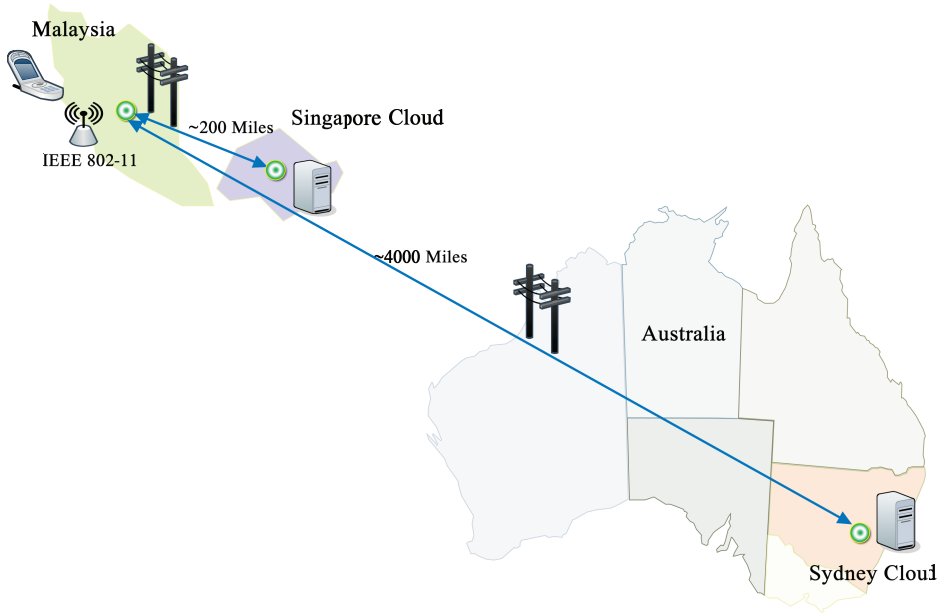



Figure 3.4: Geographical Locations of Mobile Client and Cloud Servers in the Experiment

graphical distribution of the mobile client and cloud servers in this benchmarking model. The proximate cloud which is a country-level resource is an Amazon EC2 cloud in Singapore which is approximately 320 KM away from our site in R & D Center, University of Malaya, Kuala Lumpur, Malaysia. Sydney cloud region of Amazon EC2 in Australia is selected as a distant cloud which is about 6500 KM away from the site.

On top of the hardware layer of each server, several components are deployed that are described as follows.

**Communication Handler:** Communication Handler is the communication interface of the cloud side VM that receives ajax computing call from the client and forwards the request to the Virtual Machine Manager (VMM) for resource allocation. Upon completion of the execution, the communication handler receives the response from the Synchronizer for transmission to the

client device.

**Virtual Machine Manager:** Utilizing Cloud servers is not possible without a VMM or Hypervisor. This component creates a layer over the hardware and host operating system to provide virtual computing infrastructures to the service consumers in form of virtual machine. In this experiment, VMM is essential to receive the computing service from the Amazon Elastic Computing vendor.

**Virtual Processing Unit:** Virtual Processing Unit (VPU) refers to the virtual processing elements of cloud VMs. The actual computing is performed with the help of this processing entity. Its performance consistency is remarkably important in accuracy and reliability of the analysis.

**Service Directory:** This cloud-side storage maintains the list of available service in the VM. Since we have used SOA in design and implementation of our testbed, the services are essential to be stored in the VMs. The services will be retrieved for each execution inside the VM.

**Synchronizer:** The Synchronizer is responsible to communicate with the corresponding component in the client device to maintain integration between the native code in client and the execution in the server. It is responsible to forward the computing results from the server as response and forward it to the communication handler so that it can be sent to the mobile device for reintegration. Without synchronizer component, the application's data integrity might be violated.

**Time Logger:** Time Logger server component maintains the temporal cost of executing resource-intensive component of RMA in the remote servers.

It consists of an internal timer that generates temporal data for each successful execution in the VM. These data are collected to determine the network delay by subtracting the total time from the server time to ensure the fulfillment of agreed SLA.

**Wireless communication access point:** The wireless communication access point used in this experiment is a Cisco Linksys WRT54GL Broadband Router connecting smartphone to the Internet via 2.4 GHZ band of 802.11 technology for accessing cloud servers.

### *3.2.1 (a) Implementation*

Two RMAs are implemented and executed using three different execution strategies of native, proximate cloud, and distant cloud. We have chosen math RMAs (considering noticeable increase in m-math and m-learning applications), namely prime (verifies that the given number is prime) and matrix multiplication (multiplies two $[n * m]$ and $[m * p]$ matrices). Prime RMA represents the compute-intensive group of applications. It receives one prime number and verifies if it is prime or not. Hence, the communication is negligible while the computation is intensive. The matrix multiplication application represents the second group of RMAs which is both compute-intensive and data-intensive. It receives two matrices and multiplies them to produce their product. The matrix product is a compute-intensive including large iteration of multiplications and addition. It needs both CPU and RAM to complete. It is also data-intensive tasks since it receives two large size matrices and response by one larger matrix as the product. The size comes from both the matrix dimensions and numerical values of the matrix members.

Prefabricated services are utilized to develop the experiment model and applications by programming varied platform-independent services that perform computation-intensive

tasks. These services can be executed both in mobile and cloud resources independent from the platform and device (platform-independence is one of the major characteristic of SOC architecture). While executing such intense processes in our RMAs quickly drains the mobile battery, noticeable amount of energy can be saved when utilizing cloud resources for execution. Unlike majority of offloading approaches that migrate the codes from the mobile device, we do not migrate any code from the mobile device to the cloud to reduce system complexity and only perform an http call to an already available service of the cloud and transmit data from the device to the cloud for execution (codes are not transferred, but already available web services are called at execution time). Upon successful remote execution, the results are returned back to the device. Thus, the excess overhead of identifying, partitioning, and offloading code from the mobile device to the cloud is mitigated and we can dedicate our analysis to the impact of distance between mobile device and cloud (Shiraz et al., 2012).

In programing stage, JQuery mobile version 1.3.1 framework is used on top of JQeury 1.8.0 JavaScript library. JQuery mobile is a lightweight (less than $32KB$ footprint), cross platform, JavaScript library suitable to build lightweight, online/offline, Ajax-based application for both desktop and mobile devices. Ajax (Asynchronous Javascript and XML) is a stack of several programming techniques that enables development of asynchronous client-server applications. The benefit of exploiting asynchronous client-server communications is that during execution, the components can communicate with each other in background so that users can continue interacting with the application. While waiting for the response, user is still able to interact with the client component, if any. In this experiment, each application has a client component built using HTML5 and JavaScript and a server component built using PHP version 5.3.6.

Using ROA, server-side services are called with the least messaging overhead consid-

ering the Representational State Transfer (REST) communication style deployed in ROA. REST architectural style is opted, because it generates very small computation and communication overhead due to its remarkably small header size compared to the Simple Object Access Protocol (SOAP) architectural style (Mizouni, Serhani, Dssouli, Benharref, & Taleb, 2011; Christensen, 2009). The server component in the cloud is running on Apache Tomcat web server. The server receives one request at a time and will not receive anymore request till the operation ends and response is sent back to the client. In local execution, both client and server codes are stored in the device and executed locally. Thus, the client component can send a GET or POST request to the server component available at localhost address and receive the response accordingly. To execute the PHP server code on mobile device, we install the PHP plug-in version 0.7 provided for the Android devices. To avoid connectivity issues and overhead during local execution, we run the server as a localhost without IP address by switching off the IEEE 802.11 radio from the mobile device. Mobile device in this experiment is not moving and cloud servers are selected to be immobile resources.

This experiment is performed for 30 different sample workloads (input) selected based on three computation intensity levels of low, medium, and high which are summarized in Table 3.1. The execution process for each value is repeated 30 times to ensure consistency and reliability of acquired data. Three metrics, namely overall application execution time in millisecond, network latency (in case of cloud execution) or I/O latency (in case of local execution meaning that both client and server components are running on the same device) in millisecond, and total energy consumed by each application execution in millijoule, are monitored and measured for total 900 iterations (30 iteration for each of the 30 sample workloads). The network or I/O latency values in this study exclude computation latency of server (i.e., mobile or cloud). During the experiment, the cellular radio is

Table 3.1: Description of Workloads Selected in This Experiment

| RMA | Intensity | Workload Selection Range |
|---|---|---|
| Prime | Low | $200003 \leq x \leq 290011, step \simeq 10000$ |
| | Medium | $600011 \leq x \leq 690037, step \simeq 10000$ |
| | High | $900007 \leq x \leq 990001, step \simeq 10000$ |
| Matrix | Low | $[10 \times m_i].[m_i \times 10], m_i = m_{i-1} + x, x = 10, m_0 = 0, 1 \leq i \leq 10$ |
| | Medium | $[50 \times m_j].[m_j \times 50], m_j = m_{j-1} + x, x = 10, m_0 = 110, 1 \leq j \leq 10$ |
| | High | $[100 \times m_t].[m_t \times 100], m_t = m_{t-1} + x, x = 10, m_0 = 110, 1 \leq t \leq 10$ |

switched off, USB cable is disconnected, and display brightness is set to 50%. The battery level is maintained between 60 to 70% to acquire data in more homogeneous environment and all other applications are shut down to avoid any possible interruption. We perform remote data collection after office hour during off-peak network communication hours.

To avoid man-made mistakes in collecting execution time, we exploit a auto-logging timer to start right before sending the request and to stop right after receiving the request. It is notable that execution time does not include user interaction delay for input and output. We automatically feed stored input data of an array before starting the counter to avoid data entry delay. The timer stops before results are displayed to the user. Energy consumption of processor core, main memory, and the communication chipset are monitored and other components such as LCD are discarded when collecting energy data. Energy consumed by other software components are not considered in this data collection phase and we run the applications in active mode throughout the experiment by preventing OS from pausing or blocking the application execution. PowerTutor version 1.4[1] is used to monitor consumed energy on mobile device where we parse its log reports to extract consumed energy of the applications. We analyze and interpret our data using IBM SPSS 21 product[2] that are presented in next section.

[1] http://ziyang.eecs.umich.edu/projects/powertutor/#overview

[2] http://www-01.ibm.com/software/analytics/spss/

### *3.2.1 (b) Results and Discussions*

In this section, we present our benchmarking results and discuss major finding in two parts of time and energy analysis for both applications executed based on three different execution strategies. We aim to determine the impacts of mobile-cloud distance on augmentation performance. In time analysis part, we present descriptive statistics of execution time and communication latency of executing mobile applications on varied execution strategies. Results are plotted on few bar charts and discussed consequently. Similarly, in energy analysis part, descriptive analysis of energy consumptions of mobile device is presented for all execution strategies of prime and matrix applications. We present our important findings about energy efficiency of CMA-enabled mobile applications with the help of few charts.

We perform the `tracert` command from our mobile device in the experimental site to identify the number of intermediate hops between our testbed platform and the clouds. We observe 14 hops from our place to the proximate cloud in Singapore and 23 hops to the distant cloud in Sydney. The tracing delay is about 34 and 277 milliseconds from the mobile device to the Singapore and Sydney cloud VMs respectively. The average data volume transfers in the prime application is 189 bytes which is small compared to $615KB$ of data transferred in matrix multiplication application. Although the amount of data is transferred in both experiments are small, the difference is significant enough to demonstrate the impact of communication latency, especially when the distance between mobile and cloud is long. We present our analysis results and describe the results of executing both applications on the local device with proximate cloud in Singapore and distant cloud in Sydney as follows.

**Execution Time:** We present benchmarking results of execution time of prime and matrix applications in mobile device, proximate cloud server in Singapore, and distant cloud in Sydney in Table 3.2. Also Table 3.3 presents descriptive statistics (minimum, maximum, and mean value of 900 executions) of computation time for both applications. Each number in the minimum and maximum columns in the Table represents the mean value of 30 executions of the respective workload. Descriptive statistics clearly depict the computation differences between prime and matrix applications. From the numbers in this Table, the benefits of CMA on prime and matrix applications are evidence.

Results of the prime application executions testify that augmenting resource-constraint mobile device leveraging proximate and distant cloud resources can respectively reduce the average application execution time by 75.8% and 53.1% compared to local execution. Whereas, benchmarking results of matrix application executions depict improvement in application execution time only when exploiting proximate outsourcing could reduce the average application execution time of matrix application as much as 40.7% compared to the local execution. In contrast to the proximate cloud, leveraging distant cloud for matrix application prolongs application execution time by 32.8% in average which indicates that using distant cloud not only could not save time, but also increases the execution time. Indeed, such difference in results between prime and matrix application is due to the difference in the data transmission volume, which is sharply magnified when the number of hops between mobile and cloud increases.

Networking delays due to constraints of maximum transmission unit, and TCP congestion and receive windows are signified by growth in hop's number. Hence, number of hops between mobile and cloud is a crucial factor that noticeably impacts on the quality and performance of CMA systems, especially when the volume of data transmission is high and packet size increases.

Table 3.2: Results of Execution Time for Prime and Matrix Applications in Three Execution Modes with 99% Confidence Interval

| Workload No.* | Prime | | | Matrix | | |
|---|---|---|---|---|---|---|
| | Local | Singapore | Sydney | Local | Singapore | Sydney |
| 1 | 634(+/-)34.7 | 165.8(+/-)27.3 | 447.5(+/-)50.5 | 128(+/-)34.1 | 83.6(+/-)18.9 | 460.9(+/-)89.3 |
| 2 | 662.5(+/-)40.4 | 162.2(+/-)20.9 | 414(+/-)24.3 | 150.3(+/-)34.6 | 89.8(+/-)8.5 | 597(+/-)78.4 |
| 3 | 628.2(+/-)25.4 | 170.8(+/-)30.6 | 509.8(+/-)20 | 143.9(+/-)18 | 115.4(+/-)9 | 969(+/-)62.5 |
| 4 | 660.1(+/-)29 | 199.5(+/-)64.2 | 445.8(+/-)33 | 156.6(+/-)15.5 | 128.8(+/-)10.5 | 1607.4(+/-)476.2 |
| 5 | 689.7(+/-)34.1 | 164(+/-)7.7 | 513.7(+/-)12.3 | 168.4(+/-)12.3 | 139.3(+/-)9.7 | 1103.4(+/-)57.8 |
| 6 | 683.2(+/-)41.3 | 184.7(+/-)51 | 543.8(+/-)59.5 | 189.6(+/-)33 | 172.6(+/-)9 | 792.6(+/-)109.7 |
| 7 | 731.8(+/-)30.8 | 171.7(+/-)9 | 509.6(+/-)9.6 | 210.5(+/-)16.7 | 216.8(+/-)28.8 | 1332.2(+/-)310 |
| 8 | 750.1(+/-)34.2 | 180.9(+/-)16.5 | 455.4(+/-)46.5 | 221.5(+/-)13 | 222.9(+/-)13.6 | 1193.3(+/-)313.5 |
| 9 | 728.1(+/-)39.9 | 188.6(+/-)35.4 | 518(+/-)36 | 238.1(+/-)12.5 | 238(+/-)16.2 | 1126.9(+/-)226.6 |
| 10 | 737.9(+/-)33.9 | 191.1(+/-)15.7 | 512(+/-)14.4 | 254(+/-)18.8 | 244.5(+/-)15.2 | 2221.5(+/-)347.3 |
| 11 | 1138.5(+/-)54.3 | 269.5(+/-)19.6 | 600.5(+/-)49 | 3526.1(+/-)85.5 | 1297.7(+/-)86.2 | 6674.1(+/-)705.2 |
| 12 | 1149.4(+/-)52.5 | 306.5(+/-)26 | 578.8(+/-)27.5 | 3932(+/-)93.3 | 1529(+/-)91.9 | 6822.3(+/-)706.2 |
| 13 | 1132(+/-)49.9 | 300.7(+/-)17.5 | 579.3(+/-)34.4 | 4184.8(+/-)76.5 | 1652(+/-)93.8 | 6376(+/-)659.6 |
| 14 | 1167.6(+/-)78.5 | 319.5(+/-)28.8 | 508.3(+/-)10.4 | 4608.2(+/-)96.1 | 1673.5(+/-)45.5 | 10670.9(+/-)1010.8 |
| 15 | 1158.4(+/-)53.5 | 296.1(+/-)18.6 | 557.8(+/-)33.4 | 4927.1(+/-)63.7 | 1736.2(+/-)53.7 | 12563.3(+/-)525.2 |
| 16 | 1152.7(+/-)49.8 | 335.6(+/-)38.5 | 510.4(+/-)22.5 | 5280.4(+/-)105 | 1882.1(+/-)93.3 | 12496.8(+/-)520.8 |
| 17 | 1196.1(+/-)60.7 | 307.8(+/-)27.1 | 595(+/-)15.5 | 5630.2(+/-)120.4 | 2108.9(+/-)188.2 | 13810.6(+/-)925.5 |
| 18 | 1220.2(+/-)61 | 298(+/-)14.8 | 508.1(+/-)19.7 | 5878.4(+/-)87.4 | 2703.6(+/-)707.9 | 13941.3(+/-)641.8 |
| 19 | 1194.7(+/-)64.2 | 320.8(+/-)41.8 | 597.6(+/-)47.6 | 6291(+/-)107.7 | 2559.2(+/-)620.7 | 15214.4(+/-)693.8 |
| 20 | 1241.3(+/-)63.2 | 296.3(+/-)11.9 | 517.4(+/-)16.6 | 6657.4(+/-)154.3 | 2366.5(+/-)166.9 | 15978(+/-)916.7 |
| 21 | 1558.9(+/-)67.5 | 369.6(+/-)57.6 | 583.1(+/-)23.5 | 10313.3(+/-)285.5 | 6524.5(+/-)72.5 | 14095.7(+/-)900 |
| 22 | 1592.7(+/-)65.4 | 347.5(+/-)8.1 | 623.2(+/-)24.5 | 10266.8(+/-)171.5 | 7100.1(+/-)78 | 13941.8(+/-)633 |
| 23 | 1624.3(+/-)63.7 | 373.5(+/-)44.5 | 638.9(+/-)26.1 | 12350.1(+/-)366.3 | 7892.6(+/-)154.6 | 17919.8(+/-)4480 |
| 24 | 1629.7(+/-)62.4 | 360.5(+/-)15.4 | 617.4(+/-)24.4 | 14043.2(+/-)211.1 | 8821.2(+/-)120.9 | 17203.3(+/-)1436.4 |
| 25 | 1634.2(+/-)66.7 | 350.4(+/-)13.4 | 560.2(+/-)9.1 | 15737.2(+/-)476.9 | 9752(+/-)108.3 | 17958.8(+/-)1183.4 |
| 26 | 1627.4(+/-)57.1 | 361.1(+/-)18 | 658.6(+/-)45.8 | 15917.6(+/-)435.7 | 10598.7(+/-)142.4 | 14688(+/-)1700.2 |
| 27 | 1707(+/-)63.1 | 369.9(+/-)18.6 | 581.6(+/-)25.3 | 18380.7(+/-)406.6 | 11507.1(+/-)296.6 | 13242.6(+/-)913.5 |
| 28 | 1635.4(+/-)64.7 | 368.1(+/-)14.5 | 565(+/-)7 | 19644.9(+/-)419 | 12585.4(+/-)395 | 13785.3(+/-)923.8 |
| 29 | 1687.9(+/-)78.3 | 382.1(+/-)48.9 | 572.6(+/-)8.4 | 18210.7(+/-)2941 | 13661.3(+/-)487.9 | 13934.6(+/-)1144.4 |
| 30 | 1708.5(+/-)71 | 386.1(+/-)28.1 | 597.7(+/-)20.8 | 21037.4(+/-)337.8 | 14130.7(+/-)244 | 14536.9(+/-)1103.2 |

*Workload values are presented in Table 3.1

Figures 3.5 and 3.6 illustrate mean execution time of different workloads for prime and matrix applications in three execution strategies. Each of 30 workloads is repeated 30 times and each bar in the graphs represents the mean execution time of three workloads (i.e., mean of 90 values). Figure 3.5 illustrate significant amendments in application execution time compared with the local execution time. However, Figure 3.6 shows remarkable achievements when utilizing proximate cloud only. Data related to distant cloud shows that in small workloads running RMA on mobile device is more beneficial than executing

Table 3.3: Descriptive Statistics of Execution Time and Latency in Millisecond (ms)

| Description | Application | Execution Strategy | Minimum | Maximum | Mean |
|---|---|---|---|---|---|
| Execution Time | Prime | Local | 628.2 | 1708.5 | 1168.75 |
| | | Proximate Cloud | 162.2 | 386.1 | 283.2967 |
| | | Distant Cloud | 414 | 658.6 | 547.37 |
| | Matrix | Local | 128 | 21037.4 | 6955.954 |
| | | Proximate Cloud | 83.6 | 14130.7 | 4124.471 |
| | | Distant Cloud | 460.9 | 17958.8 | 9241.956 |
| Latency | Prime | Local | 83.6 | 110.9 | 95.347 |
| | | Proximate Cloud | 59.3 | 136 | 82.6267 |
| | | Distant Cloud | 236.1 | 424.3 | 326.73 |
| | Matrix | Local | 122.5 | 11269.4 | 3777.04 |
| | | Proximate Cloud | 82.7 | 11709.5 | 3257.243 |
| | | Distant Cloud | 460 | 13351.4 | 6657.38 |

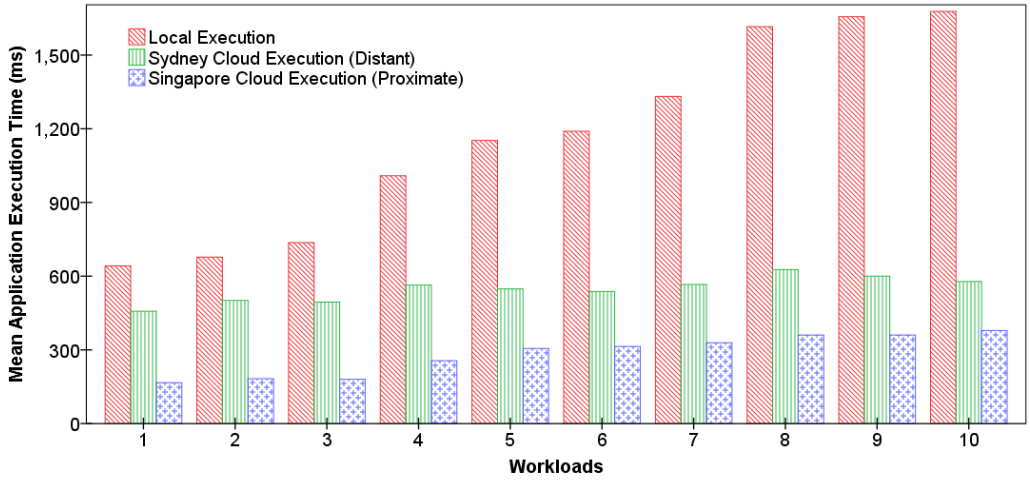



Figure 3.5: Comparison of Overall Execution Time for Three Execution Strategies for Prime Application

in distant cloud. In the synthesis of executing matrix application on proximate cloud, it can be observed that the improvement for the light workloads is much smaller than the complex workloads; execution time improvement increases when the computations intensity rises.

Although computing specifications of both cloud VMs are identical for proximate and distant clouds, the average application execution time of distant cloud is 93.3% and 124% more than proximate cloud for prime and matrix applications respectively. Such dif-

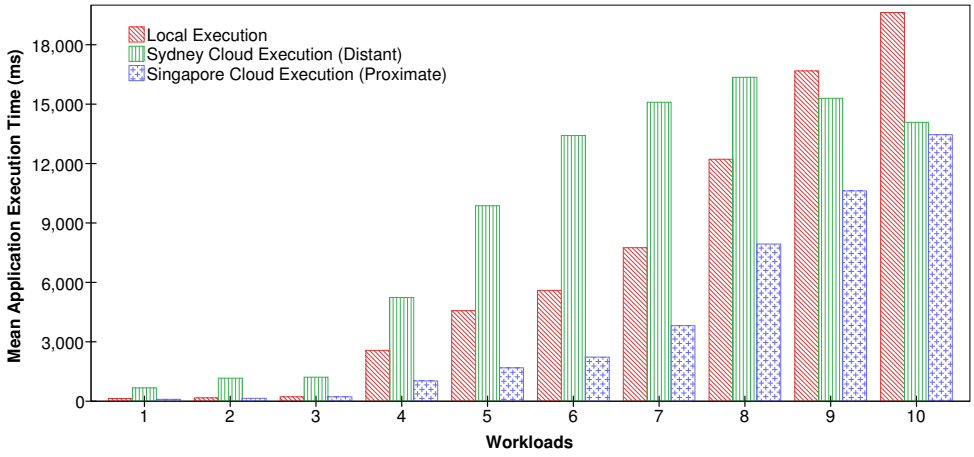



Figure 3.6: Comparison of Overall Execution Time for Three Execution Strategies for Matrix Application

ference is due to the varied number of hops between mobile and cloud, data volume, and existing heterogeneity between processing and networking infrastructures (e.g., hardware and firmware performance of intermediate nodes, bandwidth, jitter, and VM performance). Hence, the larger is the number of hops between mobile and cloud, the higher is the applications execution time.

In order to better understand the impacts of distance and hop numbers on the execution time, linear interpolation analysis is performed. The results of analysis are illustrated in Figure 3.7 and Figure 3.8 for prime and matrix applications respectively. Interpolation line of local execution in Figure 3.7 indicates sharp rise in prime execution time (low traffic) when the computing overhead of workloads increases, whereas in cloud execution environments, the time growth is lessen due to high performance cloud VMs. Indeed, it reflects the compound impacts of server's computing power, round-trip latency, and data transfer overhead of accessing the remote services on execution time. In contrast, combination of such overhead factors changes the tendency in Figure 3.8 for local and cloud execution, which is due to significantly higher complexity and data transfer overhead. Colored, varied-shape, markers depict execution time of low, medium, and high intensities workloads for both applications.

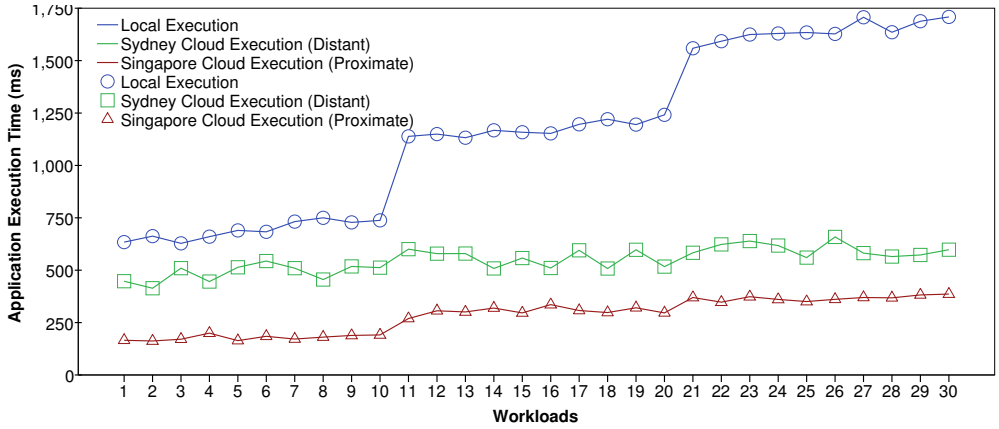



Figure 3.7: Prime application scattered plots with interpolation lines for application execution time. Predictability is feasible due to constant changes in application execution time.

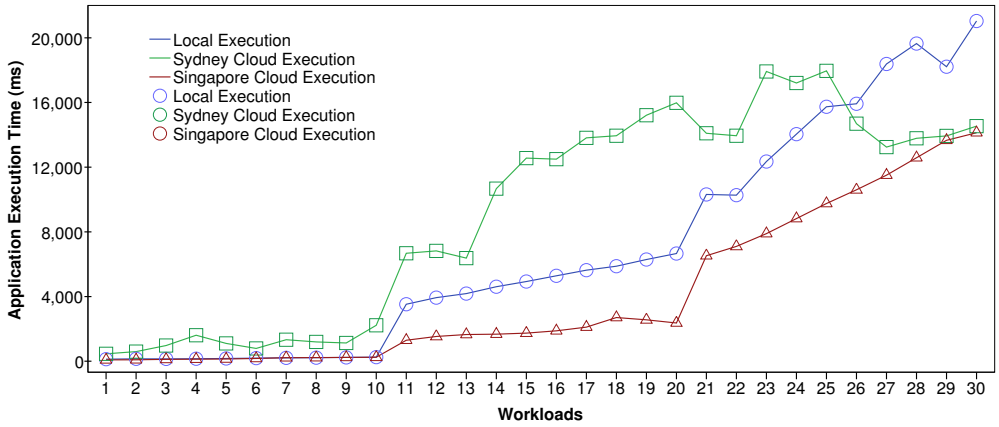



Figure 3.8: Matrix application scattered plots with interpolation lines for application execution time. Predictability is less feasible due to bursty changes in application execution time.

The observed anomaly in execution time of matrix application (see Figure 3.8) in distant cloud execution strategy encouraged repetition of data collection process for several times to ensure efficiency and accuracy of data collection and analysis methods. However, no constant change was observed during repetition which assures that predicting execution time of data-intensive compute-intensive mobile applications on distant cloud is not easily feasible. Large number of factors, particularly the momentary bandwidth fluctuations, variable jitter, climate changes, mobile resource levels, and instantaneous cloud performances are affecting the overall execution time of applications, which magnified when data volume and mobile-cloud hops increase in the presence of congestion control. Convergence of results from mathematical analysis of overall execution time and bench-

marking process advocates that in CMA systems, number of mobile-cloud hops beside several other factors such as computation intensity, mobile-cloud data transmission volume, network bandwidth and congestion, slow-start delay, and monitoring overhead of augmentation impact on the execution time of mobile applications and intensify estimation for the entire population.

Results of the prime application executions testify that augmenting resource-constraint mobile device leveraging proximate and distant cloud resources can respectively reduce the average application execution time by 75.2% and 48.4% compared to local execution. However, benchmarking results of matrix application executions depicts continuous improvement in application execution time only by exploiting proximate cloud resources (due to less number of hops). In the synthesis of executing matrix application on proximate cloud, we observe that the improvement for the first 10 light workloads is much smaller than the complex workloads; execution time improvement increases when the amount of computations increases.

Proximate outsourcing could reduce the average application execution time of matrix application as much as 36.67% compared to the local execution. In contrast to the proximate cloud, leveraging distant cloud for matrix application prolongs application execution time in average of 210% which indicates that leveraging distant cloud not only could not save time, but also increases the execution time. Indeed, such difference in results between prime and matrix application is because of the difference in the volume of data transmission that is sharply magnified when the number of hops between mobile and cloud increases. Therefore, mobile-cloud distance or especially number of hops between mobile and cloud is a crucial factor that noticeably impacts on the quality and performance of CMA systems, especially when the volume of data transmission is high and packet size increases.

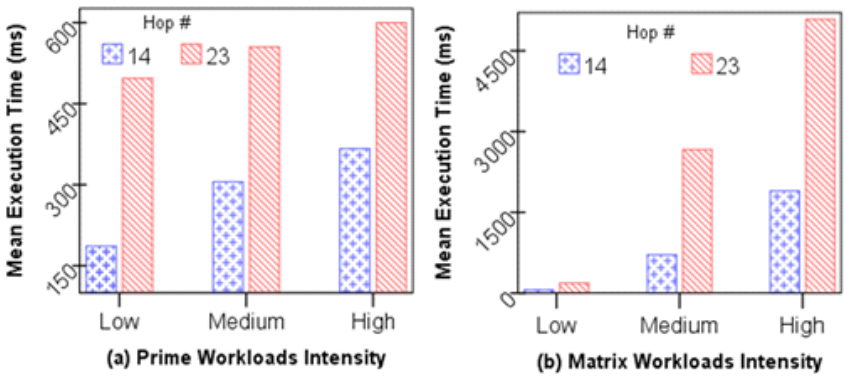



Figure 3.9: Impact of hop numbers on execution time of matrix and prime.

Figure 3.9 depicts execution times of proximate and distant modes where hop number is distinguishing metric. It helps to conclude the impacts of hops on the application performance. Each bar represents the mean execution time of ten workloads. As the bars represent, the mean execution time values are significantly larger in 23-hop cloud compared to the 14-hop cloud. These results advocate that the hop numbers significantly impact on the execution time of application. The temporal differences between 14- and 23-hop servers in Figure 3.9.(b) are more significant due to the higher transmission volume between the mobile and cloud in Matrix application.

To further highlight the impact of mobile-cloud distance and number of intermediary hops, we synthesize the round trip latency of exploiting proximate and distant cloud for prime and matrix applications. We found that utilizing proximate cloud can reduce execution time thrice than the distant cloud. Figure 3.10 depicts our synthesis results for the prime and matrix applications. Although computing specifications of both cloud VMs is identical for proximate and distant clouds, the average application execution time of distant cloud is 29% and 192.1% more than proximate cloud for prime and matrix application respectively. Such difference is due to the number of hops between mobile and cloud, data volume, and existing heterogeneity between intermediate processing and networking infrastructures (e.g., hardware and firmware performance of intermediate nodes, bandwidth, and jitter). Hence, the larger is the number of hops between mobile and cloud, the higher

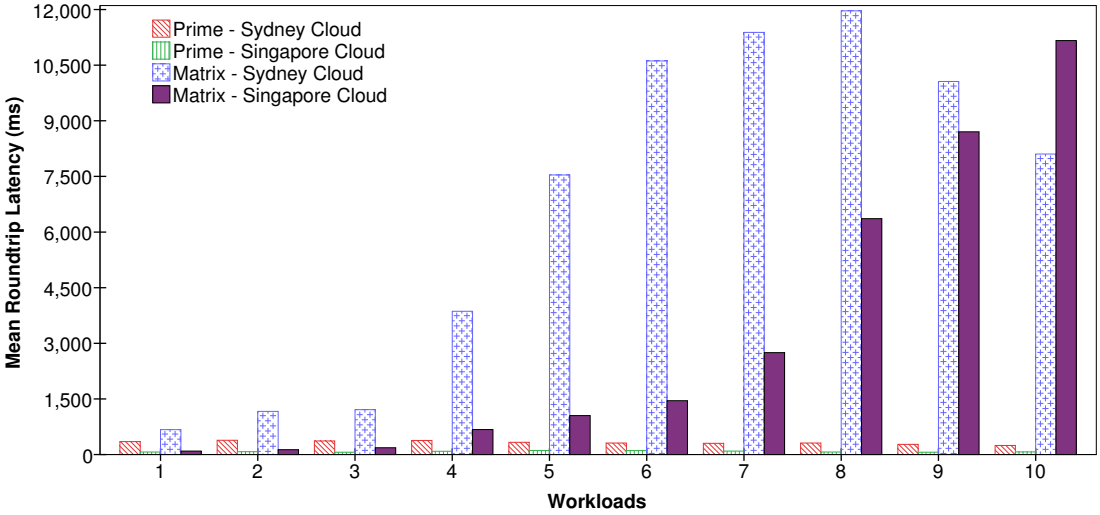
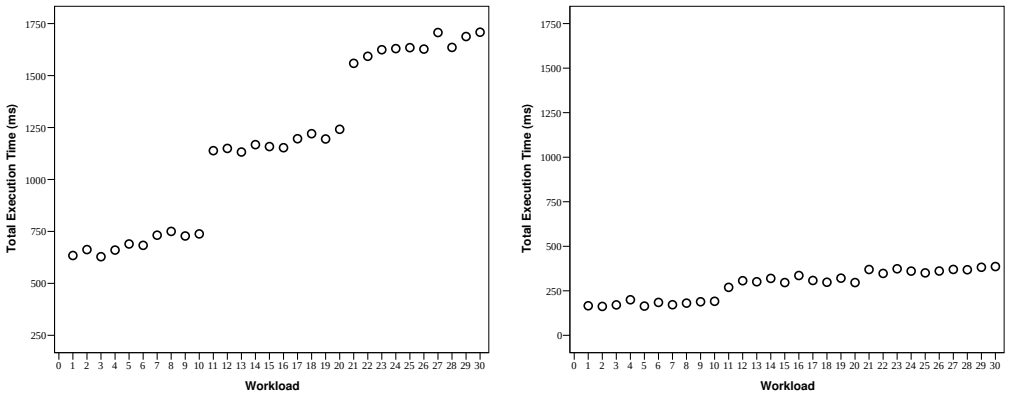



Figure 3.10: Comparison of Communication Latency of Proximate and Distant Cloud Exploitation for Prime and Matrix Applications

is the applications execution time.

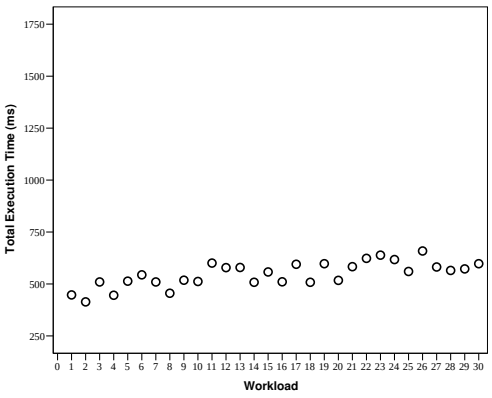
Moreover, estimation and prediction of application execution time are complicated due to the existing heterogeneity. Figures 3.11 and 3.12 depict the distribution pattern of low, medium, and high intensity workloads for both applications in our experiments. As Figure 3.11(a) illustrate, execution times of prime application are normally distributed for each of the three work intensities due to very small round trip latency (hops and data volume) between client and server components. Such constant distribution facilitates estimation and prediction of execution time for the whole population. While the proximate cloud execution distribution in Figure 3.11(b) is less scattered, distribution results of distant cloud shown in 3.11(c) visualizes more deviation from each other due to high client-server distance and larger number of hops.

Similarly, time distribution for matrix application is following a linear pattern in Figures 3.12(a) and 3.12(b) for three varied intensities. However, when the distance and number of hops between client and server components noticeably increase (see Figure 3.12(c)), the distribution pattern also diversifies complicating prediction of execution time for the entire population.




(a) Local Execution Strategy

(b) Proximate Cloud Execution Strategy

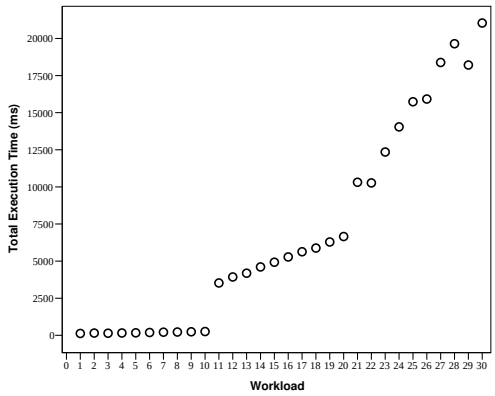



(c) Distant Cloud Execution Strategy

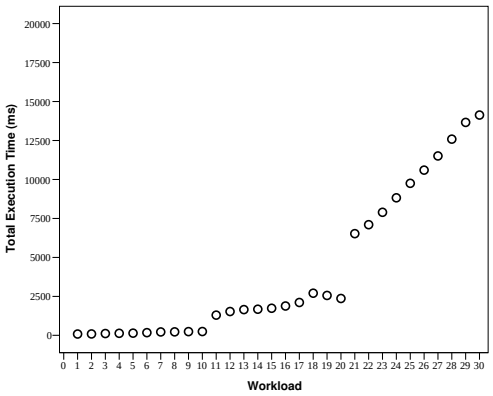
Figure 3.11: Impact of Distance on Predictability of Prime Execution Time for Three Execution Strategies

Comparison of the results from our analytical analysis of overall execution time and benchmarking process advocates that in CMA systems, number of mobile-cloud hops beside several other factors such as computation intensity, volume of data transmission between mobile and cloud, available network bandwidth, and monitoring overhead of augmentation impact on the execution time of mobile applications and degrades efficiency in RMA execution.

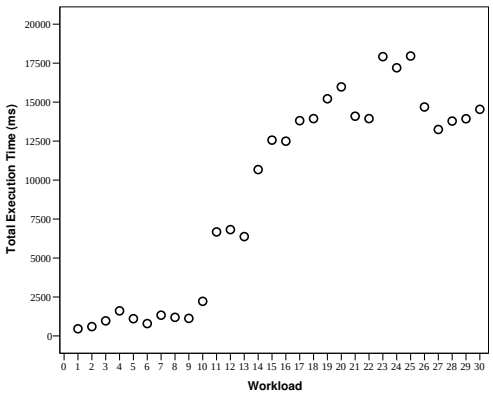
**Mobile Energy Consumption:** We present benchmarking results of mobile consumed energy collected by PowerTutor 1.4 from execution of prime and matrix applications in mobile device, proximate cloud server in Singapore, and distant cloud in Sydney in Table 3.4. Descriptive statistics of mobile energy consumption are summarized in Table 3.5. Results of prime application consumed energy advocate that augmenting resource-constraint




(a) Local Execution Strategy




(b) Proximate Cloud Execution Strategy




(c) Distant Cloud Execution Strategy

Figure 3.12: Impact of Distance on Predictability of Matrix Execution Time for Three Execution Strategies

mobile device leveraging proximate cloud resources can reduce the energy requirement of the mobile application 76.6% in average, whereas it is about 56.6% when using distant cloud. The difference in energy saving between proximate and distant cloud is significant considering the difference in number of hops and low data transfer.

Descriptive statistics for matrix application depict that exploiting proximate cloud significantly reduces the average energy consumption up to 40.6% for data-intensive mobile applications, whereas using distant cloud increases energy consumption in average by 60%. Hence, exploiting distant clouds for augmenting matrix application increases the energy consumption of the mobile application. The mean energy consumption in local mode is 4355.33 while it is 6980.39 for distant cloud.

Figures 3.13 and 3.14 illustrate the measured energy of 30 different workloads (grouped

Table 3.4: Results of Energy Consumption for Prime and Matrix Applications in Three Execution Modes with 99% Confidence Interval

| Workload No.* | Prime | | | Matrix | | |
|---|---|---|---|---|---|---|
| | Local | Singapore | Sydney | Local | Singapore | Sydney |
| 1 | 237.3(+/-)52.3 | 99.9(+/-)37.2 | 216.6(+/-)49.2 | 79.6(+/-)42.6 | 70.2(+/-)34.3 | 289.4(+/-)64.1 |
| 2 | 215.1(+/-)53.5 | 75.8(+/-)18.9 | 169(+/-)31.9 | 75.3(+/-)30.3 | 68.4(+/-)26.6 | 1072.8(+/-)467.2 |
| 3 | 230.7(+/-)49.7 | 86.1(+/-)22.4 | 181.4(+/-)17.9 | 73(+/-)27.2 | 74.1(+/-)41 | 1007.7(+/-)484.6 |
| 4 | 235.4(+/-)54.1 | 93(+/-)35.6 | 181(+/-)97.9 | 88.1(+/-)28.2 | 73.8(+/-)24.4 | 1279.5(+/-)519.3 |
| 5 | 268.5(+/-)65.9 | 100.8(+/-)37.3 | 222.4(+/-)47 | 77.9(+/-)21.1 | 105.2(+/-)27.7 | 1006(+/-)195.9 |
| 6 | 274.2(+/-)34.4 | 89.1(+/-)27.3 | 229.7(+/-)38 | 92(+/-)30.1 | 128.6(+/-)98 | 740(+/-)152.3 |
| 7 | 297.8(+/-)73.6 | 86.8(+/-)16.8 | 196.9(+/-)26.7 | 96.8(+/-)29.6 | 122.4(+/-)67.2 | 870.1(+/-)291.1 |
| 8 | 296.6(+/-)73.1 | 97.9(+/-)18.7 | 202(+/-)33.2 | 90.8(+/-)21.8 | 133.1(+/-)90.1 | 754.6(+/-)286.5 |
| 9 | 299.1(+/-)73.7 | 102.6(+/-)45.3 | 203.7(+/-)20.4 | 92.3(+/-)11.7 | 139.2(+/-)41.7 | 273(+/-)266.7 |
| 10 | 270(+/-)57.9 | 126.9(+/-)66.1 | 226.4(+/-)28.3 | 97.3(+/-)20 | 126.2(+/-)55.1 | 1576.8(+/-)357.1 |
| 11 | 509.1(+/-)102.5 | 123.4(+/-)26.6 | 244.2(+/-)41.5 | 2079.2(+/-)158.4 | 689.5(+/-)73.2 | 6297.8(+/-)444.5 |
| 12 | 601.6(+/-)108.6 | 122.1(+/-)25.2 | 274.4(+/-)45.5 | 2423.2(+/-)136.4 | 796.2(+/-)136.1 | 6044(+/-)482 |
| 13 | 556.1(+/-)94.2 | 124.9(+/-)27.3 | 261.2(+/-)59.6 | 2563(+/-)123.7 | 794.3(+/-)124.3 | 5830(+/-)553.9 |
| 14 | 617.5(+/-)96.9 | 135.4(+/-)30.5 | 250.1(+/-)51.5 | 2993.6(+/-)170.8 | 806.9(+/-)145.1 | 8072.9(+/-)411.1 |
| 15 | 608.9(+/-)136 | 124.4(+/-)35.9 | 238.4(+/-)61.5 | 3229.1(+/-)114.9 | 838.7(+/-)134.9 | 9350.1(+/-)406.9 |
| 16 | 597.7(+/-)102.8 | 145.5(+/-)34.3 | 235.8(+/-)50.3 | 3338.2(+/-)156.6 | 933.6(+/-)135 | 9491.4(+/-)461.2 |
| 17 | 589.7(+/-)97.1 | 140.4(+/-)37.1 | 272.4(+/-)64.4 | 3410.3(+/-)226.7 | 890.1(+/-)111.4 | 9786.5(+/-)460.8 |
| 18 | 630.1(+/-)122.6 | 127.5(+/-)24.3 | 242.4(+/-)50.7 | 3815.6(+/-)189.6 | 942.7(+/-)124.9 | 10192.4(+/-)343.9 |
| 19 | 612(+/-)114.1 | 148.4(+/-)36.1 | 261.7(+/-)78.2 | 3956.7(+/-)182.2 | 904.2(+/-)133.8 | 11078.1(+/-)483.9 |
| 20 | 603.5(+/-)96.6 | 132(+/-)36.9 | 273(+/-)96 | 4078.9(+/-)210.7 | 1183.3(+/-)83.5 | 11943(+/-)477.8 |
| 21 | 771.4(+/-)120.1 | 158.6(+/-)29.4 | 288.7(+/-)62.4 | 6629.1(+/-)235.7 | 4344.6(+/-)238.9 | 10612.4(+/-)778.4 |
| 22 | 753.7(+/-)145.3 | 142.1(+/-)30.3 | 302.1(+/-)79.5 | 6792.5(+/-)153.6 | 4534.7(+/-)229.6 | 10646.1(+/-)412 |
| 23 | 789(+/-)110.1 | 162.2(+/-)33.2 | 285.1(+/-)75.1 | 7218.7(+/-)271.8 | 5090(+/-)209.5 | 10806.8(+/-)542.6 |
| 24 | 812.9(+/-)148.7 | 164.2(+/-)39.2 | 274.1(+/-)30.1 | 8463.9(+/-)170.3 | 5548.4(+/-)226.2 | 11326.9(+/-)562.9 |
| 25 | 853.6(+/-)150.4 | 183.4(+/-)35.3 | 281.1(+/-)67.7 | 9935.4(+/-)266.1 | 6431.8(+/-)184.4 | 12022.2(+/-)590.7 |
| 26 | 812.8(+/-)129.8 | 158.7(+/-)43.7 | 293.2(+/-)43.1 | 9834.3(+/-)242.7 | 7174.7(+/-)227.3 | 10805.1(+/-)637 |
| 27 | 871.1(+/-)160.8 | 168.9(+/-)33.4 | 252.1(+/-)56.3 | 10845(+/-)195 | 7823.1(+/-)271.8 | 10277(+/-)520.3 |
| 28 | 838.7(+/-)156.4 | 167.6(+/-)44.5 | 258.9(+/-)58.3 | 11812.8(+/-)224 | 8205.3(+/-)291.5 | 10415.5(+/-)421.7 |
| 29 | 863.2(+/-)152.8 | 151.6(+/-)30.3 | 276.4(+/-)53.3 | 12892.3(+/-)540.6 | 8830.8(+/-)298 | 12746.5(+/-)849.2 |
| 30 | 869.7(+/-)175.4 | 189.4(+/-)29.2 | 234.3(+/-)31 | 13485.1(+/-)280.3 | 9828.9(+/-)402.5 | 12797.1(+/-)543.1 |

*Workload values are presented in Table 3.1

Table 3.5: Descriptive Statistics of Mobile Energy Consumption

| Description | Application | Execution Strategy | Minimum | Maximum | Mean |
|---|---|---|---|---|---|
| Consumed Energy | Prime | Local | 215.1 | 871.1 | 559.567 |
| | | Proximate Cloud | 75.8 | 189.4 | 130.9867 |
| | | Distant Cloud | 169 | 302.1 | 244.29 |
| | Matrix | Local | 73 | 13485.1 | 4355.331 |
| | | Proximate Cloud | 68.4 | 9828.9 | 2587.763 |
| | | Distant Cloud | 273 | 12797.1 | 6980.391 |

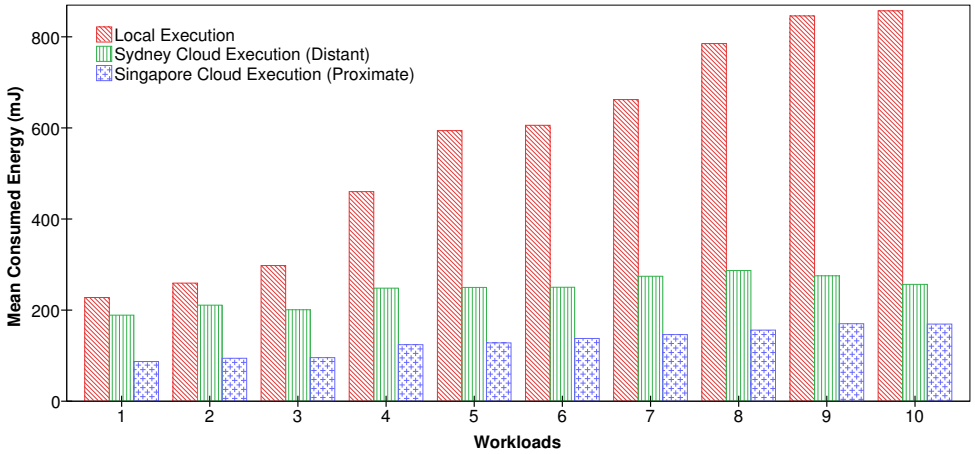



Figure 3.13: Prime application energy consumption for three execution strategies. Cloud-based execution is beneficial in all cases.

in three intensity levels of low, medium, and high) for Prime and Matrix applications respectively in three execution strategies. Each 30 workloads is repeated 30 times and each bar in the graphs represents the mean energy consumption of three workloads (i.e., mean of 90 values).

Energy consumption results of prime application indicate significant energy conservation, especially when the workload intensity increases. In contrast, in matrix application, the benefits is visible in proximate cloud only. Benchmarking results of matrix application executions depict that exploiting proximate cloud resources significantly and proportionately reduces the average energy consumption of the application up to 29.9% for data-intensive mobile applications. Whereas not only there is no improvement using distant cloud, but also there is a deterioration in energy consumption. The energy overhead using distant cloud is 3.77 times more than of its local execution that is undesirable and non-beneficial. For majority of workloads, augmenting mobile device via distant cloud results in adverse and leads to more energy consumption compared to the local execution. Energy in utilizing distant cloud is saved when the computation extensively increases and could surpass the communication overhead leading to energy saving.

For matrix application, while local and proximate execution consumed energy is rel-

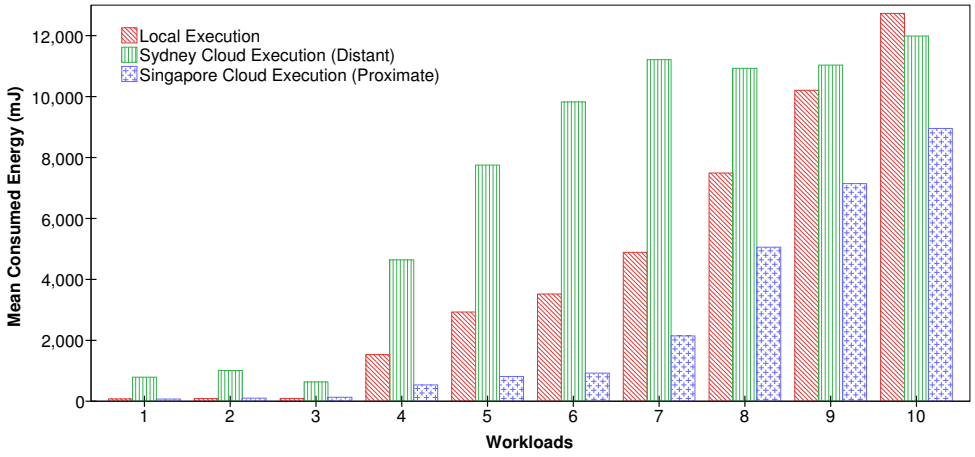



Figure 3.14: Matrix application energy consumption for three execution strategies. Cloud-based execution using distant clouds is not beneficial in most cases.

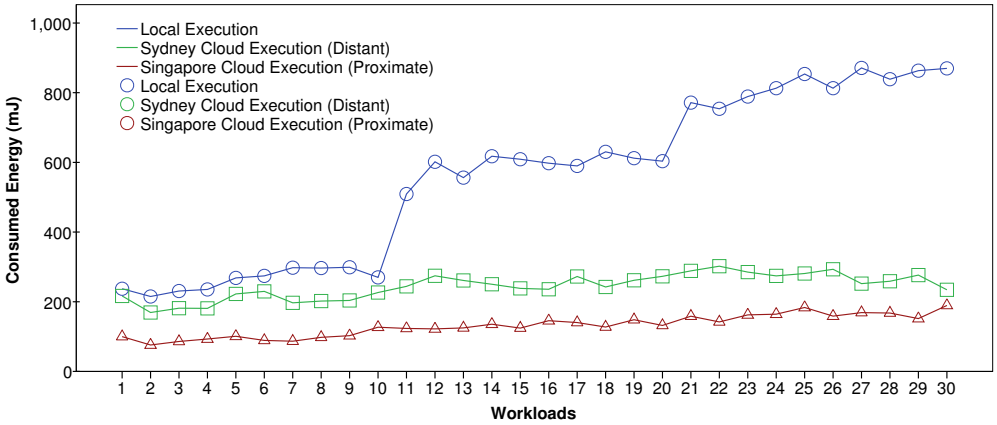



Figure 3.15: Prime application scattered plots with interpolation lines for application consumed energy.

atively steady, it is unsteady for Sydney cloud due to the overhead of long mobile-cloud distance imposed on high volume of data in blipful wireless medium. In fact, the more data transfers between mobile and cloud, the more energy consumes and the more fluctuations raises.

Similar to the time analysis, we plot the distribution pattern of the energy values using few scattered diagrams depicted in Figures 3.15 and Figure 3.16. Blue circular markers in the figures show the results of local execution mode; the red rectangular markers demonstrate the proximate cloud VM results, and the green squared markers depict the results of distant cloud VM. As can be seen, for both applications, Singapore cloud which is the proximate VM, is beneficial, whereas exploiting Sydney resources is not beneficial for

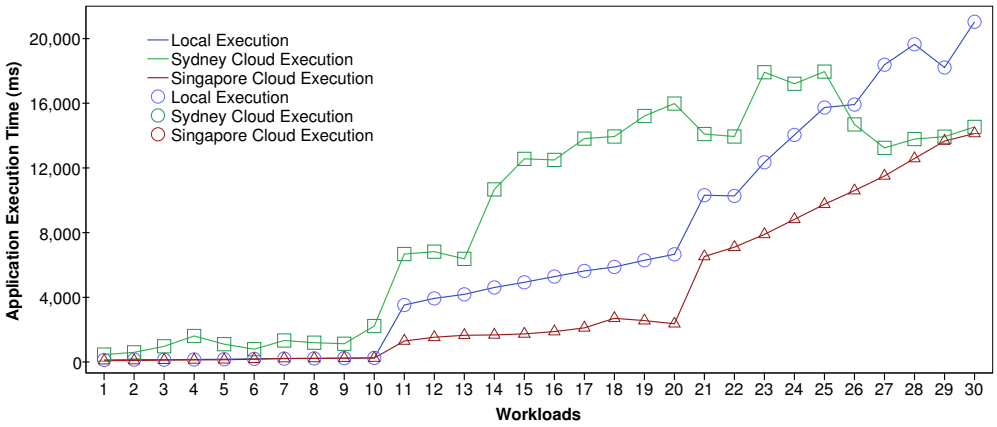
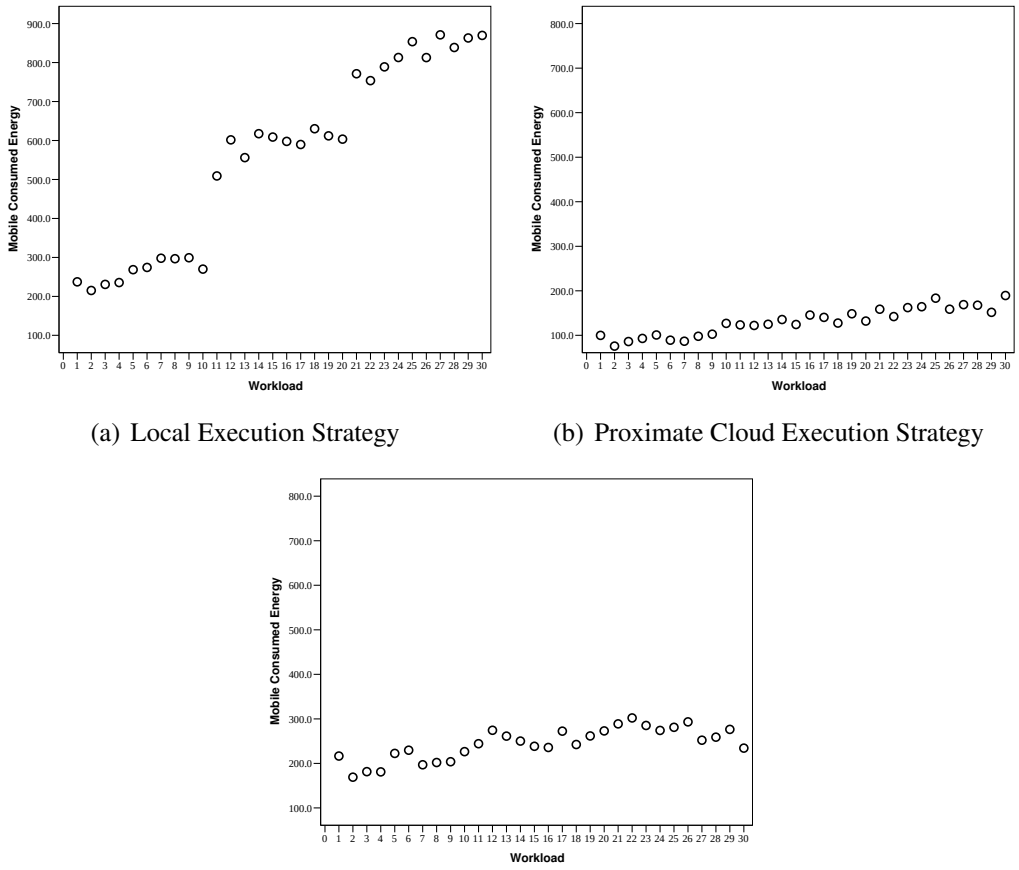



Figure 3.16: Matrix scattered plots with interpolation lines for consumed energy.

matrix application, in most of the cases. The figures also suggest unpredictability of data though it is expected to have a linear correlation between the workloads and respective consumed energy. Though the results in local execution is linear, it can not be seen in distant cloud. Such loose correlation suggests other influencing factors that are mostly caused by network intermittency and fluctuations.

To highlight the existing correlation significance, we further analyze the results which are plotted in Figures 3.17 and 3.18. In both applications, energy distribution is following normal pattern for local (chart 3.17(a) and chart 3.18(a)) and proximate execution (chart 3.17(b) and chart 3.18(b)) that facilitate estimation of energy consumption of mobile devices. However, the distribution pattern is dispersed for the distant cloud (chart 3.17(c) and chart 3.18(c)) due to the high impact of distance on the large volume of data in wireless medium. Therefore, considering the identical computing energy of cloud VMs and equal size of data transfer, intermediate hops due to long distance between mobile and cloud servers is the major factor that impacts on the energy dissipation of mobile augmentation.

We synthesize the energy consumption of exploiting proximate and distant cloud for matrix application and found that utilizing proximate cloud can reduce energy requirement of the mobile device 12 times more than distant cloud. Figure 3.19 depicts our synthesis results.




(a) Local Execution Strategy

(b) Proximate Cloud Execution Strategy

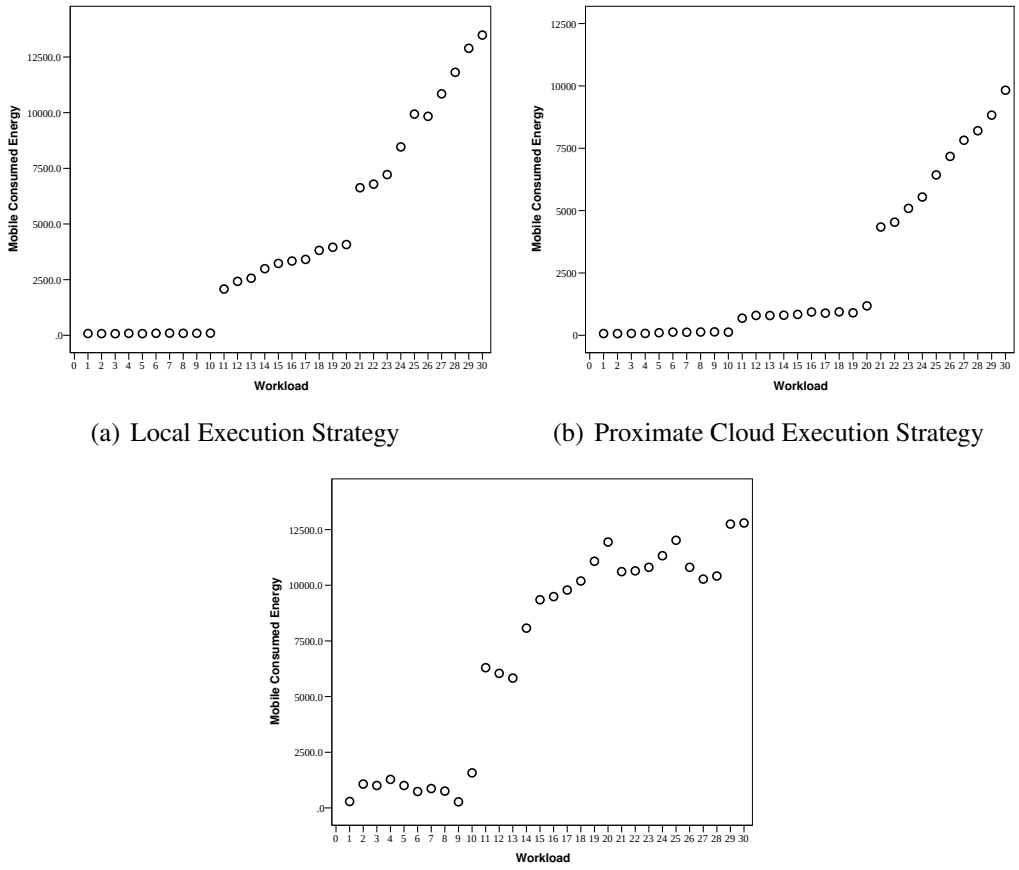
(c) Distant Cloud Execution Strategy

Figure 3.17: Impact of Distance on Predictability of Consumed Energy of Prime for Three Execution Strategies

Figure 3.20 shows energy consumed in proximate and distant modes and helps to conclude impacts of hops on application performance. Each bar is the mean energy of ten workloads. The results in Figure 3.20.(a) shows that utilizing remote resources that are accessible through 23 hops for execution prime remarkably utilizes more energy compared to of 14 hops. This difference in energy consumption is more significant in Figure 3.20.(b) due to the higher volume of data transmission between the mobile and cloud resource.

## 3.3 Conclusions

In this chapter, we analytically and experimentally analyze and synthesize the impacts of WAN latency on performance of RMAs running on resource-constraint mobile devices when exploiting geographically distributed cloud resources. Using analytical analysis on




(a) Local Execution Strategy

(b) Proximate Cloud Execution Strategy

(c) Distant Cloud Execution Strategy

Figure 3.18: Impact of Distance on Predictability of Matrix Energy for Three Execution Strategies

WAN latency we formulate round-trip delay in RMAs and found that the number of intermediary hops as well as data transmission volume significantly impact on the performance of RMAs. Using series of benchmarking experiments in real environment we verify and validate our findings. We benchmark the results of executing two RMAs, namely prime verification and matrix multiplication using three execution strategies of local, proximate cloud, and distant cloud to analyze the impact of distance and number of intermediate hops between the resource-constraint mobile clients and resource-rich cloud servers.

In this study we found that performance gain of utilizing remote resource is highly influenced by amount of data transmission between mobile and remote servers and number of intermediate hops. Our finding complement the work by other researchers in this domain (Kumar & Lu, 2010) who suggest to perform trade-of between computation and

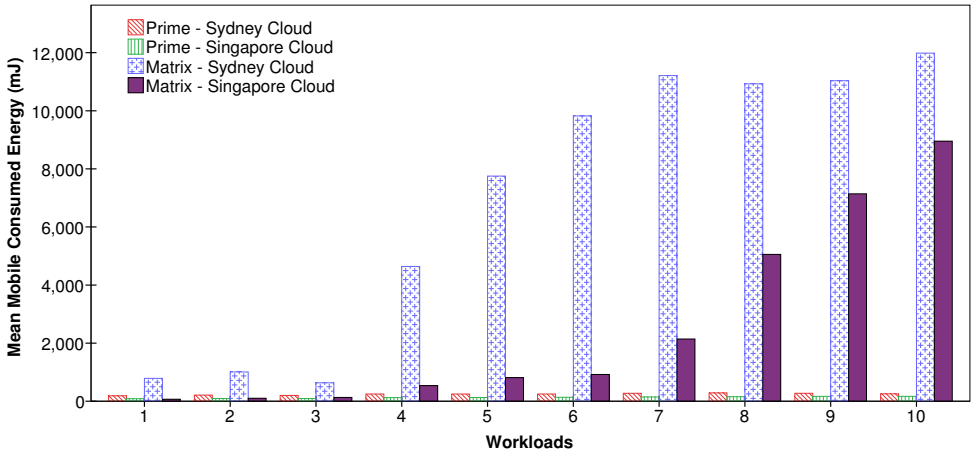



Figure 3.19: Comparison of Energy Consumption in Singapore and Sydney Clouds for Prime and Matrix Applications

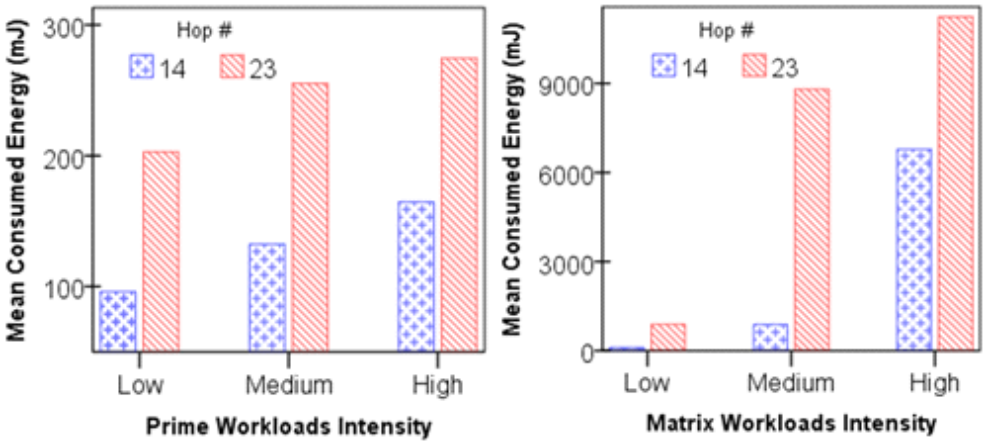



Figure 3.20: Impact of hop numbers on Energy usage of matrix and prime.

communication before offloading. Our benchmarking results advocate that exploiting distant cloud resources to augment computational capabilities of resource-constraint mobile devices is a time- and energy-intensive approach that degrades the performance of augmented mobile applications in MCC and inhibit efficient execution of RMAs in resource-constraint mobile environment.

Moreover, increase in number of mobile-cloud intermediate hops, significantly impacts on the distribution pattern of execution time and energy in RMAs. Augmenting mobile devices by leveraging distant clouds originates an anomaly and encumbers accurate time and energy estimation of mobile applications (due to large number of hops). Such impact is increased when data communication volume between mobile and cloud raises.

Therefore, if offloading data-intensive operations to remote resources is unavoidable, reducing the number of intermediate hops between mobile clients and cloud servers significantly enhances the performance of resource-intensive applications on resource-constraint mobile devices toward efficient execution of RMAs.

# CHAPTER 4

# LIGHTWEIGHT MOBILE CLOUD COMPUTING FRAMEWORK

This chapter presents the framework proposed in this research. With the help of few schematic diagrams we present the main building blocks and components of the proposed framework and describe their functionality. Moreover, the interaction among major components of the framework is illustrated using sequential diagrams and described in detail. The design of data to evaluate the performance of our work is also presented and the statistical methods and test required to analyze and synthesize the data are described.

The remainder of this chapter is as follows. Section 4.1 presents the comprehensive description of the proposed framework. Section 4.2 describes the major building blocks of the proposed framework. Significance of the proposed framework is presented in section 4.3. Section 4.4 identifies data to be generated from the framework and explains how to generate required data. Chapter is concluded in section 4.5.

## 4.1 Lightweight Mobile Cloud Computing Framework

We proposed a lightweight MCC framework based on ROA that is capable of augmenting resource-constraint mobile devices in efficient execution of RMAs in MCC. ROA is the SOA design philosophy (Guinard, Trifa, & Wilde, 2010; ?, ?) based on RESTful architectural style (Fielding, 2000). Designing our framework based on ROA has the following benefits according to the existing research (Christensen, 2009; Richardson & Ruby, 2007; Guinard et al., 2010).

- Portability: Designing our framework based on ROA design philosophy provides portability of applications to varied platforms with least configuration and modifi-

cation.

- Loosely Coupling: Designing framework based on ROA enables loose coupling of the framework and the applications that are built based on the framework. Loose coupling of applications contribute to lightweight characteristics of the framework by omitting the partitioning overhead of tightly coupled applications.

- Statelessness: Statelessness in SOA mitigates resource consumption of SOA-based applications in the absence of need to store the service states (Erl, 2008). Statelessness that leads to conserving native mobile resoruces is critical in compute-intensive mobile applications due to resource deficiency of mobile devices.

- Autonomy: Applications build based on ROA deliver high level of control over the running environment leading to improved runtime behavior predictability which is vital in MCC. The more control a service-based application has over its underlying runtime environment, the more predictable its runtime behavior would be (Erl, 2008). Predictability of applications are necessary in deciding whether to offload execution of a service/functionality from mobile device to remote resources or not. Unpredictability leads to inaccurate offloading decisino making that causes inefficient execution of RMAs.

- RESTful: Design and implementation of our framework using RESTful architectural style reduces the communication overhead compared to SOAP (Mizouni et al., 2011).

- Cacheable: REST and SOA provide caching mechanism to the RMAs. Provisioning caching mechanism in design and development of applications enables mobile servers to store the response even after disconnection from wireless network. With-

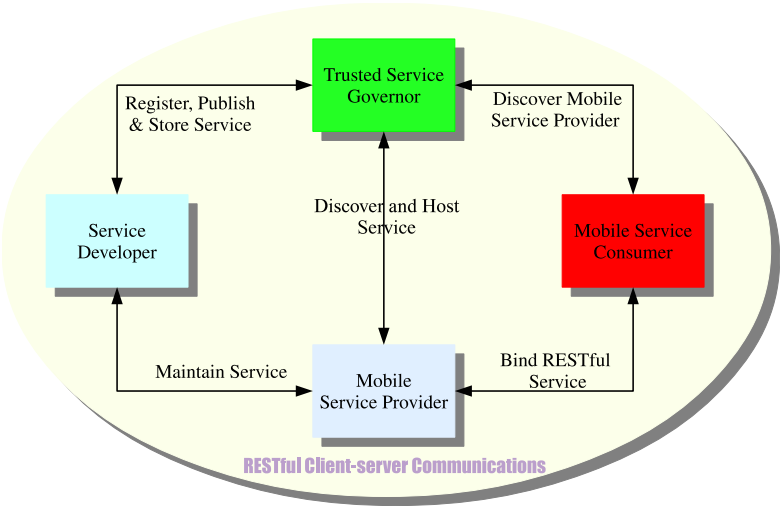



Figure 4.1: The Block Diagram of the Proposed Framework

out caching, the entire computation must be repeated that increases energy consumption and degrades energy efficiency.

Our proposed framework consists of four major building blocks, namely Service Developer, Trusted Service Governor (TSG), Mobile Service Provider (MSP), and Mobile Service Consumer (MSC) depicted in Figure 4.1. An abstract description of the system and its operational tasks are briefly described as follows and the in-depth description is presented in section 4.2.

Initially, the Service Developer initiates a task and registers, stores, and publishes its ready-to-bind service(s) to the TSG (which plays the role of Universal Description Discovery and Integrity (UDDI) for Web services). In the meantime, mobile device owners contact TSG to learn the list of available services that can be hosted and executed on their mobile devices; once found, device owners acquire service code and register with the TSG as potential MSP to host the service for future execution.

RMA developers use the registered services to build complex RMAs which can be acquired by mobile end-users to execute on their mobile devices. Mobile end-users who are exploiting computational resources of MSP are named MSCs in this research. During

RMA execution, the MSC contacts the TSG to identify an appropriate MSP who can provide required services. Upon successful MSP discovery, the MSC can bind with the MSP service and utilize the available service(s) based on pay-as-you-use concept of cloud computing (payment can be zero or more depends on the service delivery model). To ensure service accuracy and durability, Service Developer has direct channel with the MSP to maintain and update the service(s) when required.

### 4.1.1 Separation of Responsibilities

In design and development of this framework we leverage the separation of responsibilities to enhance the adoption of the framework in real practice; in traditional ROA systems, service providers have two responsibilities, namely service development beside service hosting and execution. Service developer implements or purchases service implementation, supplies its description, and provides technical and business maintenance, hosts the service, and executes it by itself. However, to enhance our framework adoption, feasibility, and complexity, we separate the responsibilities by substituting "service provider" with two independent entities of Service Developer and MSP. The former is responsible to build, describe, and maintain the service, while the latter hosts and executes the service. Below, we further elaborate role of each one in detail.

**Service developer:** is an organization or individual programmer responsible for design and development of specific fine-grained utility service that will be hosted on plethora of heterogeneous mobile devices, including smartphones, tablets, and notebooks to offer operation to the MSCs. Service developers are able to publicize their services to be consumed by service end-users. Service complexity varies from a small mathematical function to a complex task and also mashup of several small services can lead to complex services. However, services should be lightweight building blocks that are executable

on contemporary mobile devices with least possible footprint. Hence, resource-intensive services are required to be developed using mashup of fine-grained sub-services to avoid runtime code migration and offloading. However, invoking overhead is an important factor while defining service granularity because invoking very fine-grained services imposes processing and communication overheads that may prolong overall execution time leading to service consumer's resource drainage.

In order to accommodate resource-intensive services whose execution needs giant computing resources, cloud resources are provisioned in design and development of our framework. Intensive services that are not executable on nearby smartphones, will be hosted and sent to the cloud providers for which users should pay as they use based on a Service Level Agreement (SLA).

Service developer negotiates with TSG building block to describe and publish its services. During negotiation, metrics such as developer reputation, service usefulness, and price are evaluated by the governor to accept service registration. Upon mutual conclusion, the service is registered with the governor. In order to fulfill lightweight property of the proposed framework and to mitigate overhead of matching services, the clustering approach is deployed in design and implementation of the tables and database. Each table in the database is clustered based on particular service types, such as math, algebra, geometry, and image processing. So, at the time of registering new services into the system, appropriate service type is selected for efficient and accurate clustering.

**Mobile Service Provider (MSP):** According to the separation of responsibilities deployed in design and development of this framework, the MSPs play the role of hosting and executing already-developed services on-demand. The MSP can contact the TSG to identify the services suitable for execution on its mobile device. Upon conclusion, the service code is installed in the MSP device for future execution requests. Although, we

have separated these two roles, still owners of the mobile devices who host services can be service developers and vice versa.

## 4.2 Building Blocks

We illustrate the main building blocks of the proposed framework using a schematic model consists of major components of each building block in Figure 4.2. Structural and behavioral characteristics of each component are explained as follows.

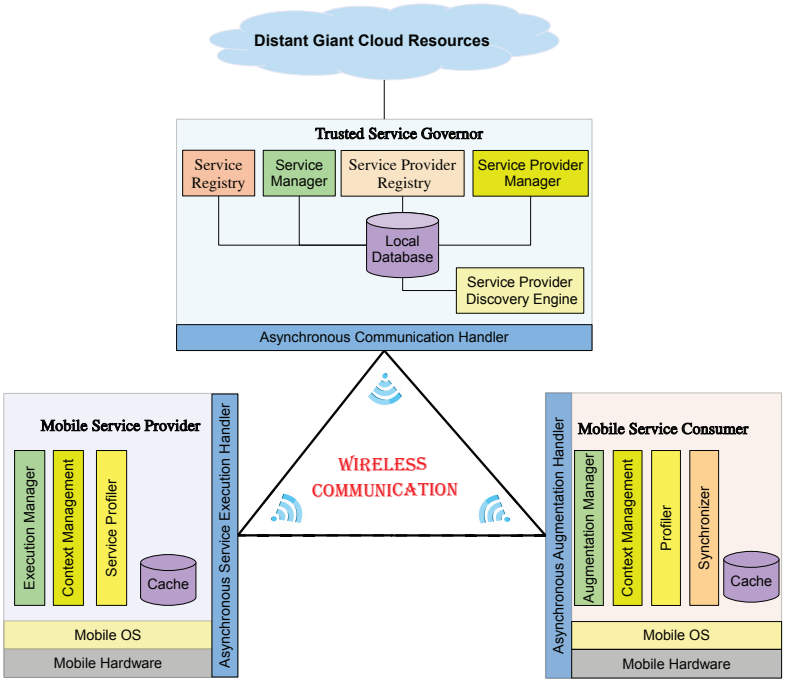



Figure 4.2: The Systemic View of The Proposed Framework

### 4.2.1 Mobile Service Consumer (MSC)

Mobile service consumers (MSCs) are the mobile end-consumers of the prefabricated services. The mobile end-users consume these services at application runtime to augment their device processing abilities and conserve local resources by exploiting computation capabilities of cloud of nearby MSPs and cloud datacenters (if needed). Prior execution, the MSC can configure the application to be executed whether locally or remotely using

our framework. In finer level of control, users are capable of enabling remote execution of individual intensive services, if desired. Following components with very lightweight footprints (less than 10kb) are essentially developed to be installed on MSC devices for efficient utilization of the system.

Pseudo codes presented in Algorithm 1 are the abstract level operations of the MSC to leverage the computational resources provided by our framework. For execution of each service $s_i$, the code checks if the service belongs to the list $J$ of resource-intensive category. For all resource-intensive services, the framework utilizes the PMC resources for efficient execution. Non intensive services are executed locally.

- **Asynchronous Augmentation Handler (AAH):** This component is responsible to asynchronously initiate remote connection with the external computing devices, including TSG and MSPs. The request is designed to be sent asynchronously so that mobile user can keep interacting with the device while the request is responded by the external server. The AAH receives remote execution request from the augmentation manager and asynchronously forwards the request to the TSG to identify the IP address of the appropriate MSP. Once found, the TSG forwards the IP addresses to the MSC through the AAH which sends the IP addresses to the augmentation manager for further processing. At remote execution time also, the AAH receives the execution request from the augmentation manager and communicate with the MSP to complete the remote execution task.

- **Augmentation Manager:** Augmentation manager is responsible to ensure seamless execution of RMA over remote servers via managing and monitoring the entire augmentation procedure. The execution of RMA starts from local device and cloud-based augmentation is initiated when the local execution reaches a resource-

intensive service(s) that is/are not executable on the mobile device (or its local execution is not preferred).

**Algorithm 1:** RMA's Service Execution Using Proposed Framework: Pseudo Code of MSC Operations

```
Data: service name, user data, user requirement
Result: Efficient RMA Execution
Begin;
for each service do
    if service s_i ∈ J then
        Fetch(context) ;   // Fetch context data from context management
        TSG − URI = Resource ID of the TSG;
        Ajax(URL: TSG − URI, Name:s_i, Requirement: Requirement[...]) ;
        // asynchronously contact TSG
        if success then
            //binding info is received as response;
            Service − URI=response;
            Ajax(URL:Service − URI, Data: UserData);
            if PMC execution succeed then
                Synchronize data;
                PMC execution of s_i completed;
            else
                Handle Error; either retry PMC execution or terminate execution
        else
            Handle Error and retry;
            Try Local Execution Mode if PMC mode not possible;
    else
        Ajax(URL: localhost − URI, Data: UserData) ;      // local execution
        if Local execution succeed then
            Integrate data;
            local execution of s_i completed;
        else
            Handle Error; either retry local execution or terminate execution
RMA Execution Finished;
End.
```

In this stage, Augmentation Manager receives context information (e.g., user location) from Context Management component and initiates an asynchronous post call to the Service Provider Discovery Engine inside the TSG building block looking for an appropriate

MSP in vicinity. Considering the user preferences, input parameters can either be sent at service discovery call or later when calling the MSP. In the former, the TSG forwards the request along with the input data to the nominated MSP for execution. Upon successful execution, the results can be either returned to the TSG for security verification -if the security is preferred- or directly forwarded to the MSC (required URI is sent by the TSG).

In the latter case, upon successful MSP discovery, the TSG pushes back the URI(s) of all the corresponding nodes (when more than one service are searched) to the MSC. The retrieved resource identifier(es) is used by Augmentation Manager to build Unified Resource Identifiers (URIs) of the resource in the nominated MSP(s). Hereafter, the services are called for remote execution and results are re-integrated to the RMA using Synchronizer component. While the execution takes place in remote server, MSC can keep performing normal local tasks until the results are received. Since the communication overheads (energy and time) of transmitting input data to TSG and MSP is the same, we deploy the latter case while implementing and evaluating our framework.

Moreover, Augmentation Manager monitors the communication link of the MSC to maintain/reestablish the connection if the MSC loses the communication. Additionally, results are cached locally to be used when the connection is restored. Hence, energy efficiency and responsiveness of the MSC is improved and system robustness is enhanced.

- **Synchronizer:** Synchronizer component maintains data integrity at online mode when augmenting computing power of MSC via cloud-based augmentation. During the remote service execution, all update requirements take place via synchronizer component in the background without notifying end-users. The operation of the Synchronizer is critical to ensure the data integrity of the RMA while performs using remote execution.

- **Context Management:** Context management acquires and maintains required context information about the MSC including its physical location. The context information are used by the Augmentation Manager to more accurately identify the proximate MSP. The user location is determined by the unique International Mobile Subscriber Identity (IMSI) number that is generated by the MNO and is stored inside the Visitor Location Register (VLR) database. Using the Location Area Identifier (LAI) value inside the IMSI number, the MNO can identify the geographical location of the mobile subscriber at any time.

- **Profiler:** During augmentation process, the profiler component inside the MSC is responsible to log QoS metrics, such as execution time and provider's reliability upon every successful/failed execution of consumed service(s). The log file of the service can be forwarded to the TSG after each execution. However, it degrades the system lightweight property due to overhead of frequent communications from MSC to the TSG. In order to reduce such communication overhead and enhance energy efficiency of the mobile application, the collected QoS metrics of the application is stored on MSC and is forwarded to the governor when user prefers. These metrics and historical information are leveraged in ranking of web services. In future, the system is able to select the best ranked service for each client inquiry via performing service ranking mechanism.

- **Dual Caching Mechanism:** Because both mobile service providers and consumers are communicating via an unstable wireless network, we deploy dual caching strategy for enhanced performance. Accordingly, we consider two local caches; one on the MSC and another on the MSP to enable them to cache data when disconnected. The cached data will be transferred to the correspondent upon reconnection.

### 4.2.2 Mobile Service Provider (MSP)

A Mobile Service Provider (MSP) is a mobile device, particularly smartphone, tablet, laptop, and car-mounted computer which is able and desired to host the prefabricated services and execute them for MSC on-demand. If a mobile owner desires to host a service and run it on behalf of a resource-constrain MSC, the MSP needs to register with the central supervisory entity (i.e., TSG) and upon approval install the MSP components on its device. The MSP components run on top of the mobile OS and consists of five major components, namely asynchronous service execution handler, execution manager, context management, service profiler, and cache. Similar to the MSC, we consider a local cache in this framework to enable devices to cache response at disconnection state. Major components are explained as follows:

- **Asynchronous Service Execution Handler:** Asynchronous listener is designed to listen to incoming requests from the TSG and MSC to initiate/perform service execution. In addition, transmitting the response to the destination is handled by this component. All communication are taken place using HTTP protocol. Similarly, the overhead of continuous polling for response is mitigated by employing push mechanism and forwarding the response as soon as the message is prepared using asynchronous communication. Asynchronous nature of communications in this framework enables continuous interaction of the end-user with the mobile device that enhances quality of user experience and improves RMA execution efficiency.

- **Execution Manager:** Execution manager receives the requests from Communication Handler, initiates service execution, monitors successful service execution, receives context information from the Context Management component, and forwards response from the service to the Communication Handler according to the latest con-

text information. One of the most important roles of the execution manager component is to monitor the wireless link of the MSP. In case of any disconnection, it not only caches the results but also tries to establish secondary communication link to forward the results to the MSC. Such task significantly improves energy efficiency and responsiveness of the augmented mobile device since network disconnection or interruption does not necessitate discovering new MSP and performing separate call to another service provider.

- **Service Profiler:** Service profiler can monitor QoS metrics such as resource requirements, reliability, and availability of every service that is executed in MSP to facilitate and optimize appropriate QoS ranking of various services. Execution time is logged using a counter that counts during execution. Reliability and availability data are historical information collected upon successful/failed execution that can be used to rank service availability and reliability. Such logged historical information are sent regularly to the TSG to ease ranking so that unreliable services are omitted from the service database. Because considering resource limitations of both service providers and consumers, failure and inefficiency of services are not tolerable. Similar to the profiler component in MSC, the log file are sent after certain number of executions (based on user preference) of each service to avoid energy wastage of the MSP.

- **Context Management:** One of the most profound advantages of utilizing mobile devices as host is their perception capability. Context management component is responsible to ensure that required context information is available. The MSP location can be monitored by the MNO using the IMSI number that is stored inside the mobile device and MNO database. This component performance is crucial for the

adoption of the framework since the MSP should be the closest mobile node to the MSC to tackle communication overhead.

### 4.2.3 Trusted Service Governor (TSG)

Trusted Service Governor (TSG) is a trusted supervisory entity replicable to be located on multiple servers which are responsible for supervision and monitoring of the entire augmentation system, including MSP, MSC, and service developers. In abstract, major building blocks communicate to each other via this building block, though partial direct communications are also provisioned (e.g., developer and MSP for service maintenance). Considering huge number of services, developers, hosts, and requesters, it is vital to provision a supervising and monitoring entity for the success of the whole system. TSG is the main governing entity in proposed framework with several crucial responsibilities that are depicted in Figure 4.2 and described as follows. To avoid overcrowded servers and to mitigate the impacts of long WAN latency, our feasible and beneficial solution is to replicate TSG components on as many server as possible on different geographical areas. For the optimal performance, the TSG layer needs to be replicated on all MNOs so that MSCs with different network operator can utilize the service continuously

Considering the supervisory role of MNOs, their reputation-based trust, and latest emerging trend as cloud service provider, the most appropriate servers to host the TSG layer is deemed to be MNOs. MNOs have been serving mobile end-users from the beginning and could establish reputation and historical trust. They are scattered in urban and rural areas near the MSP and MSC. Their proximity to the mobile nodes significantly reduces the WAN latency which is crucial for mobile augmentation. MNOs have started providing cloud services to their clients and, hence, will be able to scale and adapt to the highly oscillating computation and storage of end-users. Moreover, replicating TSG on

MNOs aims to alleviate the challenges of mobile web service provisioning. Therefore, the framework employs MNOs as the computing hosts for the TSG layer. The components of TSG block are described as follows.

- **Service Registry:** Service registry acts as a public service repository similar to UDDI. Service registry maintains a local database called 'service database' to store full description of registered services. At the time of service registry, service developer negotiates with TSG and provides detailed information about the service. It also uploads the complete service package including core and dependent libraries. These codes will be later downloaded by MSPs who are willing to host these services. The sequential diagram of the service registry process is depicted in Figure 4.3. Because the service codes are designed to be executed inside multitude of mobile devices, codes need to be stored inside a central publicly available storage.

- **Service Manager:** Service manager analyzes historical QoS information that regularly receives from the MSP and MSCs for periodic efficiency assessment leading to efficient evaluation and ranking of the overall functionality and performance of various services. Service developer is responsible to maintain the service in case of any malfunction and optimize its performance based on collected data in service database. Service Manager enhances quality and reliability by continuously analyzing the overall performance of every individual service to substitute low quality services with more efficient ones –if required.

- **Service Provider Registry:** Every potential MSP communicates with the central service registry to browse available services and select the most appropriate services to host. The MSP must choose the services with less resource requirement than its available resources. For example, a service with 512 MB memory need cannot be

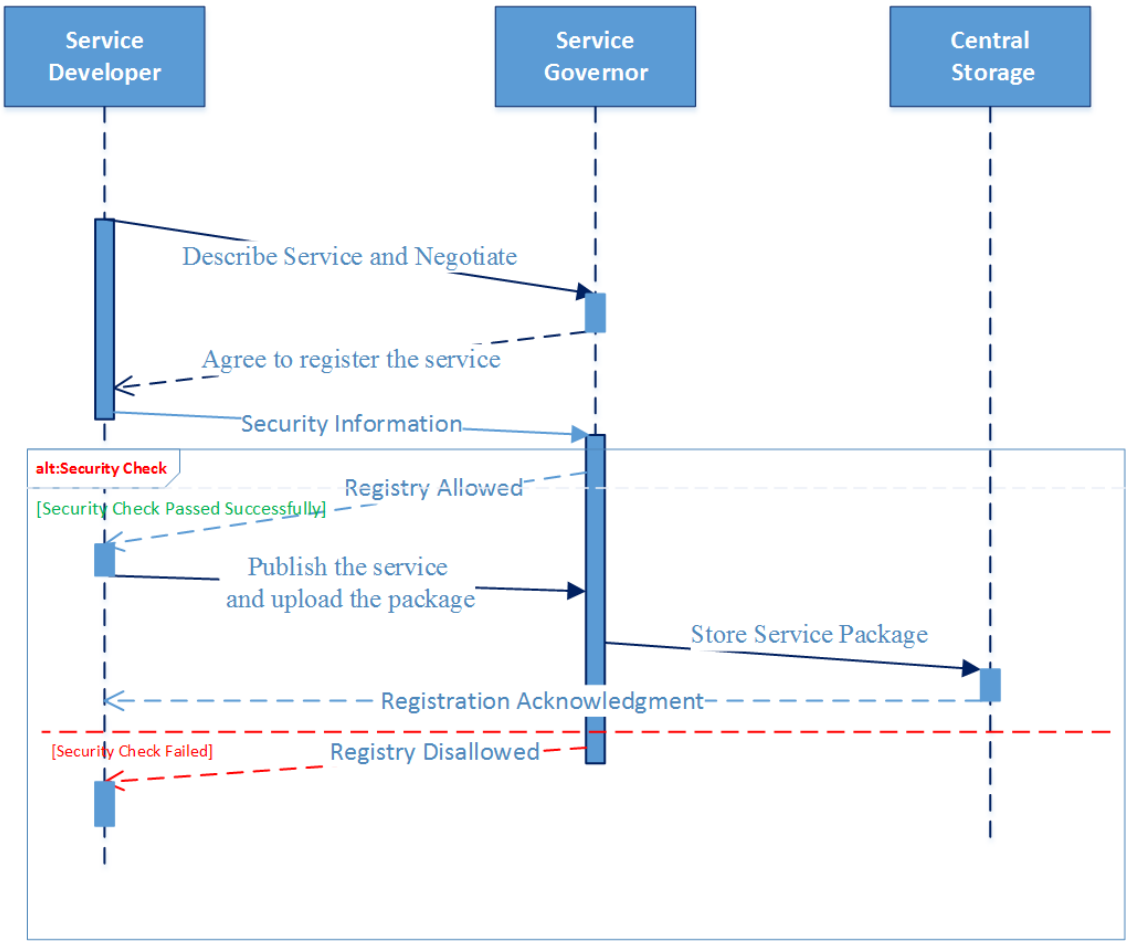



Figure 4.3: Sequence Diagram for Service Registry

hosted on a device with 256 MB memory. The Service Provider Registry component is responsible to validate the hosting demands and refuse inappropriate allocation requests. During registry, the governor and service provider negotiate and establish trust. Upon successful registry, a unique Uniform Resource Name (URN) is issued and allocated to the services hosted on legitimate MSP.

- **Service Provider Manager:** Similar to the Service Manager component that evaluate services, Service Provider Manager also analyze and synthesize historical QoS information of all MSPs and rank them accordingly based on their performance, availability, and reliability. These ranks are considered when Service Provider Discovery Engine searches for MSP of a particular service. At selection time, the TSG choose the MSP with highest rank.

- **Asynchronous Communication Handler:** Communication handler is one of the major components of the TSG in this framework. This is a communication interface for Service Registry, Service Manager, Service Provider Registry, Service Provider Manager, and Service Provider Discovery Engine where requests are handled and verified by the request handler component. Valid requests are forwarded for further processing to the corresponding components and invalid request are dismissed. Once a valid request is forwarded to the corresponding component, the communication handler maintains the communication link in the background to return the response. All components in our framework are employing asynchronous communication to retain the lightweight property, enhance energy efficiency, and enrich user interaction experience.

- **Service Provider Discovery Engine (SPDE)**

  We further enhance lightness of our framework by levying the service discovery overhead on the TSG (which is a non-mobile device) instead of the MSC. Service Provider Discovery Engine (SPDE) is responsible for determining the most appropriate MSP capable of performing MSC's requested service(s) with desired preferences. Initially, MSC makes an asynchronous call to the SPDE component of the TSG and forwards service name(s) and preferences (e.g., operating system name) along with the request to inquire and acquire the URI address of the best nearby MSP capable of providing desired service(s) and satisfying delegated preferences. For instance, if the MSC needs the factorial value of X, it sends the operation name or its description (require semantics to interpret the description) along with user preferences.

  SPDE receives the inquiry from the MSC and searches the entire database of ser-

vices to validate the service name. Once service name is valid and requested service can be provided, we extract the MSC's geographical location from the VLR database that is maintained and stored in the MNO (the SPDE is hosted in MNO). Considering the user location and current location of MSPs, SPDE analyzes the database to identify the most appropriate MSP that can efficiently meet MSC requirements and preferences. If only one provider that fulfills all requirements exists, the SPDE forwards its URI to the MSC. But, if more than one MSP found, the one with higher reputation/mark is selected. If SPDE does not have access to the MNO registry of VLR, the MSC can forward its locally stored IMSI to the SPDE for location identification.

Leveraging database clustering mechanism enhances search time, because at the time of service matching, the system only searches among certain cluster(s). Clustering is proven to significantly shrink the search time. For discovering services in each cluster, keyword-based approach is leveraged (semantic description requires semantic-based matching approach). Initially services are matched using keyword-based model (i.e. based on service type, service name, and operating system (e.g., service type is math, service name is prime, and target OS is Android)) which is feasible and beneficial solution, because developed RMAs are already utilized known services of our database. Therefore, the keyword-based search system can promise optimized performance from temporal cost point of view.

Figure 4.4 shows sequence of collaboration among major building blocks in this framework when running a RMA. A resource-constraint MSC forwards an asynchronous request to the TSG asking the required service(s) along with its preferences and requirements. The TSG determines in the first stage whether the requested service exists or not.

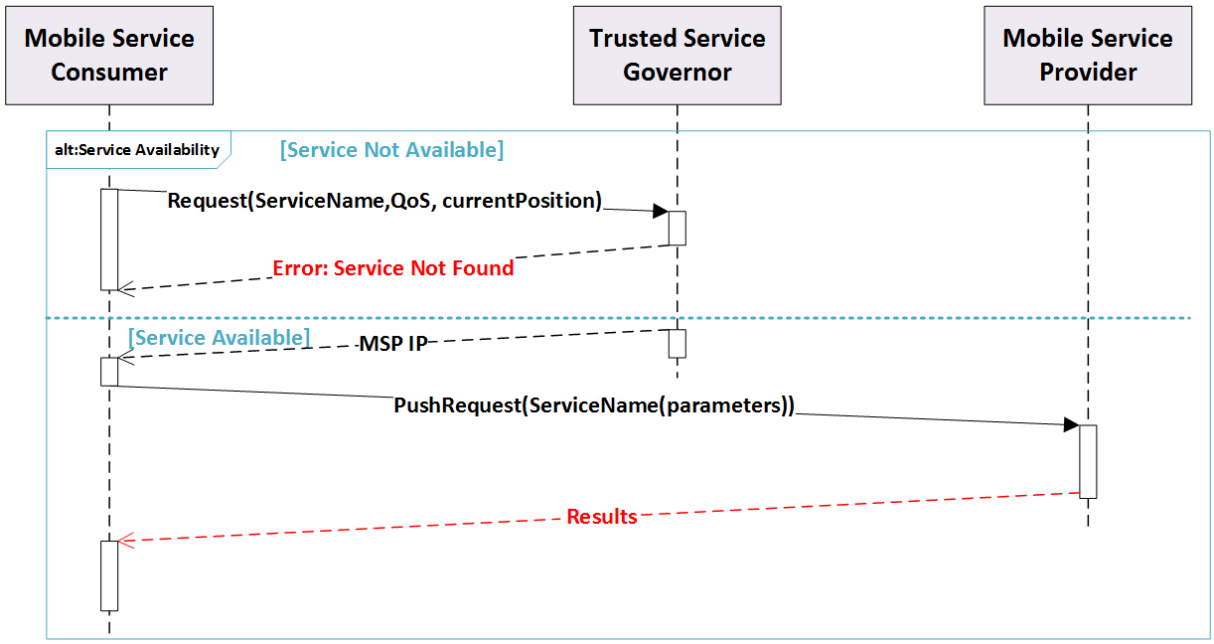



Figure 4.4: The Collaborative Scenario Among Major Entities

If not, the execution terminates. If the required service(s) exists in the database, the TSG replies with the IP address(s) of the potential MSP and the URI of the desired service. The MSC contacts the MSP and forwards input data for processing. While the MSC is waiting for the response, the end-user can continue interacting with the device. The MSC decides whether bear the cost of security breach and receive direct response from the MSP or ask TSG to verify the response and forward the safe response to it. In this research, we consider direct communication, because MSPs are considered secure.

The flow of operation to perform complete mobile empowerment process in our framework is depicted in Figure 4.5. The left box is this Figure shows the component and operations of the TSG at runtime whereas the right box in the chart illustrates the operations in the MSP. The flow starts from the top most *start* state and ends at the bottom most *end* state.

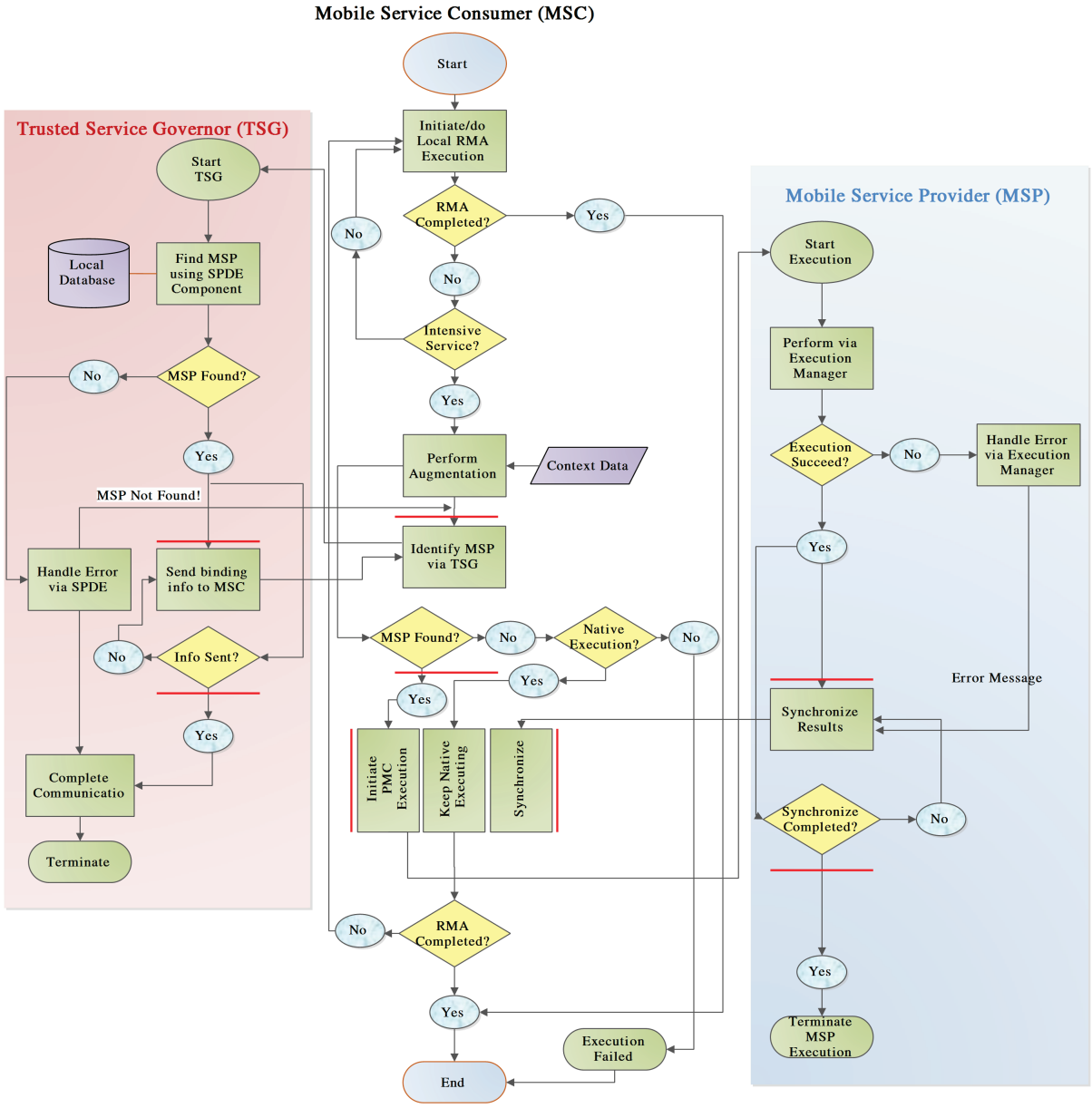



Figure 4.5: The Flow of Mobile Empowerment Operation in Proposed Framework

### 4.2.4 Wireless Communications

In this framework, due to existence of wireless MSPs and MSCs, we require to perform wireless communication. Varied wireless communication technologies, mainly cellular Wireless Local Area Network (WLAN), or Bluetooth can be employed in our framework. However, utilizing cellular communications must be minimized since cellular networks are energy-hungry (Perrucci et al., 2011). Provisioning of heterogeneous

communication technologies, including cellular network enables us to incorporate user mobility in our framework. Once the MSC user moves out of the serving MSP coverage area, the communication can continue using cellular network, instead of terminating communication and repeating augmentation.

### 4.3 Significance Features of the Framework

Several key features that are considered in design and development of this framework to enhance its significance and novelty are briefly discussed as follows.

- **Lightweight RMAs:** RMAs that are built according to this framework inherit attributes of ROA, particularly loose coupling of services with the rest of the application and virtualization-free execution on heterogeneous mobile devices. Augmenting execution of service-based applications in MCC originates significantly lower overhead compared to highly coupled codes in non-service-based applications. Augmenting non service-based applications necessitate partitioning and separating the intensive codes from the application prior the remote execution and also demands reintegration of the results after completion of the remote execution process. Thus, the RMAs deployed based on this framework are lightweight in nature.

- **Platform-independence:** Leveraging ROA in design and implementation of this framework makes this solution independent from the underlying platform. So, any mobile device including smartphone, laptop, tablet with any operating system including Android, iOS, Blackberry, Windows mobile, Windows, and Linux can leverage the benefits of this proposal with least possible overhead. ROA not only mitigates the deployment cost of this framework in different operating systems, but also omits the overhead of utilizing virtualization which are effective in achieving RMA execution efficacy.

- **Non-obtrusive Interaction:** Non-obtrusive interaction refers to the feature of mobile distributed applications/systems that avoids freezing of the application while performing client-server communications and enables continuous and distraction-free interaction between the application and end-user leading to significant improvement in quality of user experience. We achieved non-obtrusive interaction by employing asynchronous communication offered by Ajax technology in design and development of our framework. In asynchronous communications, the client-server communication and data transfer take place in the background and hence the end-user is capable of interacting with the device in the foreground.

- **Internet-free Communication:** Unlike majority of MCC solutions that require to connect to distant cloud datacenters through the Internet, this framework establishes the connection with the MNO or nearby MSPs without need to pass through the risky insecure channel of Internet. Apart from security hazards of communicating via Internet, utilizing Internet service to accessing cloud resources, originates long WAN latency and levies billing cost of data plan on the user that decrease usability of this framework in reality.

- **Centralized Architecture:** In order to reduce the management complexity, shrink the MSP discovery overhead, and enhance security of communicating with insecure MSPs, a central reputed authorizing entity called TSG is designed. By deploying the service provider discovery engine into the TSG, the overheads of maintaining the system functionality and discovering the MSP are transferred from the MSC to the TSG. Moreover, TSG can be equipped with techniques to control the authenticity and reliability of MSPs at registration time and throughout the life time. For instance, TSG can ask MSPs at the time of registration to prove their personal iden-

tity which discourages them from attacking the MSC or falsifying the results.

- **Short WAN Latency:** Leveraging cloud of proximate mobile devices as MSP significantly reduces the communication and WAN latency of accessing remote resources. Mobile devices are the most proximate ubiquitous computing devices to MSCs. Hence, utilizing such pervasive mobile devices decreases WAN latency.

- **Non-voluminous Communication Overhead:** In our proposal, we neither partition the intensive code nor send the code to the TSG or MSP. The intensive code are already available in the TSG and MSPs. Hence, there is no need to migrate the code to the MSP at execution time. Considering huge number of calls to such computational codes (services), the communication overhead and wireless traffic are significantly decreased.

- **Mobile Service Provider:** Separation of responsibilities and enabling mobile device to serve as service provider without having technical/programming knowledge and expertise is another significant feature of this proposal. Traditional SOC-based systems entertain three main roles of service provider (including code developer), service consumers, and service broker. However, in our model four major roles are provisioned, including service developers, mobile service consumers, mobile service provider, and trusted service governor. Using this new approach, owners of mobile devices can simply browse the TSG's database and select the service to host on their mobile device without having programming and development skill.

- **Green Mobile Computing:** One of the most important characteristics of this framework is its approach toward green mobile computing. This framework enables owners of off-the-shelve or faulty-yet-functioning mobile devices to share computing resources of their devices either for free or in exchange of certain credit (either

reputation or finance) if billing system is incorporated into the system. Considering huge number of rapidly antiquated mobile devices, this approach can save the environment by more efficient utilization of old devices.

## 4.4 Data Design

In this section, we describe the methods used to evaluate the performance of this framework. We also identify the performance evaluation metrics that are identified for evaluation of the framework.

### 4.4.1 Evaluation Metrics

We identify application execution time and consumed energy as two metrics of evaluation that are presented in Table 4.1 and explained as follows. These two metrics help us to obtain our aim and objectives in this study. Consumed energy and execution time are the most important established metrics to evaluate the lightweight properties of a typical MCC framework.

- **Consumed Energy:** Consumed energy is the amount of energy consumed to complete the entire life cycle of one prototype RMA for one workload which is measured and stated in milliJoule presented by mJ.

  The energy data is collected using Power Tutor 1.4 which is capable of collecting energy data for the computing and wireless communications. The energy consumed by the other components such as storage and LCD are disregarded in this study. In order to avoid man-made mistakes, the energy values are extracted and acquired from the log files created by the Power Tutor and processed using regular expression techniques.

Table 4.1: Performance Metrics for Performance Assessment

| Metric | Definition | Unit | Symbol |
|---|---|---|---|
| Execution Time | Total time takes to complete one entire life cycle of a single workload | Millisecond | ms |
| Consumed Energy | Total amount of energy consumed on the device to complete one entire life cycle of a single workload | MilliJoule | mj |

- **Execution Time:** Execution time is the amount of time taken to complete the entire life cycle of one prototype RMA for one workload which is measured and stated in millisecond (ms).

  Different methods of generating execution time exist including manual and automatic data collection. Similar to the energy, in order to avoid man-made mistake and to obtain accuracy, we have developed automatic logging timers that start counting at the beginning of the application execution and ends when the application successfully completes.

## 4.5 Conclusions

In this chapter, we illustrate the proposed framework in this research and described its characteristics. The schematic presentation of the proposed framework and its major building blocks are depicted. We also described our proposed framework by explaining functional and non-functional characteristics of its main components. Sequence diagrams are used to outline the functional behaviors of major components of the framework. Moreover, several significant features of the framework are highlighted. Data design is described and data generation procedures are explained in detail. We have identified application execution time and consumed energy as the performance evaluation metrics. Benchmarking and statistical modeling are determined as the performance evaluation and validation techniques. The data analysis methods that are utilized in evaluation of this framework are presented.

# CHAPTER 5

# EVALUATION

This chapter presents the performance evaluation methods used to evaluate and validate the execution performance of our proposed framework. To evaluate the proposed model and its lightweight features, we use benchmarking experiments for application execution time and consumed energy. Using 30 synthetic benchmarks, we collect and analyze application execution time and consumed energy data. The evaluation results are validated via statistical modeling. To build our statistical model, we used independent replication method and validate the proposed model using split-sample approach. In another set of experiments, we build a separate test-bed with mobile devices different from the main performance evaluation experiments and prototyped the applications using ASP (Active Server Page) as a different language and employed Sieve benchmarking (as an standard benchmark algorithm) and a new set of benchmarks to demonstrate that the execution performance of our framework is not biased by any particular mobile device, programming language, benchmarking algorithm, and even special benchmarks. Moreover, we performed a comparative study of our application development and intensive-code execution mechanisms used in our framework with the offloading mechanism that is dominantly used in the literature, including (Chun et al., 2011; Satyanarayanan et al., 2009) to demonstrate the lightweight feature of our framework. Lastly, we describe the statistical data analysis methods used to analyze and synthesize the results.

The remainder of this chapter is as follows. Section 5.1 presents the benchmarking experiments along with data generation methods. We describe statistical modeling and data generation methods in section 5.2 and explain how the statistical models are vali-

dated. In Section 5.3, we explain the benchmarking experiments used to demonstrate that our framework is platform-independent. Section 5.4 presents our comparative study and Section 5.5 describes the statistical data analysis methods used in this thesis to analyze and synthesize the results. Chapter concludes in section 5.6

## 5.1 Benchmarking Modeling

The prototype RMA consists of three different computing services are developed that are executed on two execution modes of local and PMC for performance evaluation of the proposed framework in terms of execution time and consumed energy. We perform the computation and analysis in two execution modes, namely local and PMC execution modes. In the former, we execute the entire task locally on the mobile device without utilizing remote resources. In the latter, the RMA is started from the mobile device and execution of intensive components are asynchronously called at runtime to be executed on nearby mobile devices. The schematic presentation of our benchmarking setup is illustrated in Figure 5.1 and described as follows.

To prepare the real experimentation environment, we use a powerful desktop computer with Intel i5-2500 processor having 3.3 GHz clock speed, 4 GB memory, 1 TB storage, and 32-bit Windows 7 professional accessible via wireless connection and install the governor layer of the proposed framework on it. The MSC is a HTC Nexus One smartphone that is hosting and running MSC components of our proposed framework and contains the developed RMAs. Three mobile devices, including one Samsung I9100 Galaxy S2 smartphones, one HTC One X, and one DELL Laptop XPS14z are selected as MSP. The Samsung smartphone features Dual-core 1.2 GHz Cortex-A9 CPU 1 GB RAM, 32 GB storage, 802.11 Wi-Fi radio, and a 3.7v with 1650 mAh capacity battery running Android 4.0.3. The HTC smartphone features Nvidia Tegra 3 chipset with Quad-core 1.5


Figure 5.1: Schematic Presentation of the Benchmarking Setup

GHz CPU, 1 GB RAM, 32 GB storage, 802.11 Wi-Fi, and Li-Po 1800 mAh battery running Android 4.2.2 OS. The laptop computer features Intel i5-2450M chipset with 2.5 GHz clock speed CPU, 4GB RAM, 1 TB storage, running 64-bit Windows 7 home accessible via wireless connections.

The mobile device used in this experiment is HTC Nexus One that features a RISC 32-bit Qualcomm Snapdragon S1 QSD8250 chipset[1]. This chip has a single core ARMv7 application processor with maximum 1024 MHz clock frequency.

The wireless communication access point used in this experiment is a Cisco Linksys WRT54GL Broadband Router building a communication platform to connect MSC and MSPs via 2.4 GHZ band of 802.11 technology.

For implementation purpose, we exploit JQuery Mobile (JQM) version 1.2.0[2] (the latest stable version) which is a modern programming framework based on HTML5,

[1] http://www.qualcomm.com/media/releases/2007/11/14/qualcomm-premieres-snapdragon-first-chipset-solutions-break-gigahertz

[2] http://jquerymobile.com/

Javascript, and CSS technologies. JQM features lightweight footprint (less than 12kb compressed), and also progressive enhancement by which the framework can be cross-platform, cross-device, and cross-browser. Thus, our framework will be portable to large number of mobile devices (e.g., smartphone and Tablet) including Android, iOS, and BlackBerry operating systems. The server side codes are implemented using PHP programming language and database is deployed using MySQL engine.

We evaluate execution performance of each of the RMAs for 30 different workloads into three intensity levels of low, medium, and high that are summarized in Table 5.1. Each workload is executed 30 times whose mean value is considered for analysis to enhance the reliability of the results. The findings are presented with 99% confidence interval in this experiment to ensure data collection reliability.

To collect execution time, we exploit automatic timer to start right before sending the request and to stop right after receiving the response. It is notable that execution time does not include user interaction delay for input and output. We automatically feed stored input data from an array into the system before starting the counter to avoid data entry delay. The timer stops before results are displayed to the user.

Energy consumption of processor core, main memory, and the communication chipset are monitored when collecting energy data. Energy consumed by other software components are not considered in this data collection phase and we run the applications in active mode throughout the experiment by preventing OS from pausing or blocking the application execution. PowerTutor version 1.4[3] is used to monitor consumed energy on mobile device where we parse its log reports to extract consumed energy of the applications.

In order to ensure data reliability and consistency, during the local execution mode experiment, the cellular radio of the MSC is switched off, USB cable is disconnected, and

[3]http://ziyang.eecs.umich.edu/projects/powertutor/#overview

Table 5.1: Workloads Description: Value, Request Size, and Response Size

| Workl'd # | Intensity | Prime | Req Size Byte | Res Size Byte | Matrix Multiply | Req. Size KB | Res. Size KB | Matrix Covariance | Req. Size KB | Res. Size KB |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Low | 220063 | 576 | 418 | [30x60][60x60] | 140.3 | 17.4 | [45x45] | 52.8 | 20.3 |
| 2 | | 233071 | 576 | 418 | [30x70][70x60] | 163.8 | 17.6 | [47x47] | 57.6 | 22.1 |
| 3 | | 248267 | 576 | 418 | [30x80][80x60] | 187.2 | 17.7 | [49x49] | 62.6 | 24 |
| 4 | | 250967 | 576 | 418 | [30x90][90x60] | 210.7 | 18.1 | [51x51] | 67.8 | 25.9 |
| 5 | | 265277 | 576 | 418 | [30x100][100x60] | 234.2 | 18.4 | [53x53] | 73.2 | 28 |
| 6 | | 272777 | 576 | 418 | [30x110][110x60] | 258.2 | 18.6 | [55x55] | 78.9 | 30.1 |
| 7 | | 284833 | 576 | 418 | [30x120][120x60] | 282.3 | 18.7 | [59x59] | 87.7 | 34.5 |
| 8 | | 310027 | 576 | 418 | [30x130][130x60] | 306.3 | 18.8 | [61x61] | 97.1 | 36.9 |
| 9 | | 320009 | 576 | 418 | [30x140][140x60] | 330.5 | 18.9 | [63x63] | 103.6 | 39.3 |
| 10 | | 330017 | 576 | 418 | [30x150][150x60] | 354.9 | 19 | [65x65] | 110.3 | 41.8 |
| 11 | Medium | 600011 | 576 | 418 | [65x95][95x120] | 459.2 | 81 | [72x72] | 135.4 | 51.2 |
| 12 | | 610031 | 576 | 418 | [65x100][100x120] | 483.5 | 81.2 | [74x74] | 143 | 54 |
| 13 | | 620003 | 576 | 418 | [65x105][105x120] | 508.3 | 81.4 | [76x76] | 150.9 | 57 |
| 14 | | 630017 | 576 | 418 | [65x110][110x120] | 533.2 | 81.7 | [78x78] | 158.9 | 60 |
| 15 | | 640007 | 576 | 418 | [65x115][115x120] | 558 | 81.8 | [80x80] | 167.2 | 63.1 |
| 16 | | 650011 | 576 | 418 | [65x120][120x120] | 582.9 | 82 | [82x82] | 175.7 | 72.8 |
| 17 | | 660001 | 576 | 418 | [65x125][125x120] | 607.7 | 82.1 | [84x84] | 184.4 | 76.4 |
| 18 | | 670001 | 576 | 418 | [65x130][130x120] | 632.5 | 82.2 | [86x86] | 193.3 | 80 |
| 19 | | 680003 | 576 | 418 | [65x135][135x120] | 657.4 | 82.4 | [88x88] | 202.4 | 83.8 |
| 20 | | 690037 | 576 | 418 | [65x140][140x120] | 682.6 | 82.4 | [90x90] | 211.8 | 87.6 |
| 21 | High | 900007 | 576 | 418 | [130x60][60x180] | 490 | 236.6 | [100x100] | 261.6 | 108 |
| 22 | | 910003 | 576 | 418 | [130x70][70x180] | 572.2 | 240.5 | [105x105] | 289 | 119.1 |
| 23 | | 920011 | 576 | 418 | [130x80][80x180] | 654.4 | 242.9 | [110x110] | 317.8 | 130.6 |
| 24 | | 930011 | 576 | 418 | [130x90][90x180] | 736.6 | 244.6 | [115x115] | 347.9 | 142.7 |
| 25 | | 940001 | 576 | 418 | [130x100][100x180] | 818.8 | 245.8 | [120x120] | 379.4 | 155.3 |
| 26 | | 950009 | 576 | 418 | [130x110][110x180] | 902.8 | 246.7 | [125x125] | 412.3 | 168.5 |
| 27 | | 960017 | 576 | 418 | [130x120][120x180] | 986.8 | 247.4 | [130x130] | 446.6 | 182.2 |
| 28 | | 970027 | 576 | 418 | [130x130][130x180] | 1070.8 | 247.7 | [135x135] | 482.3 | 196.5 |
| 29 | | 980027 | 576 | 418 | [130x140][140x180] | 1155.5 | 248.2 | [140x140] | 520 | 211.2 |
| 30 | | 990001 | 576 | 418 | [130x150][150x180] | 1240.7 | 251.2 | [145x145] | 559.1 | 226.6 |

display brightness is set to 50%. The battery level is maintained between 60 to 70% to acquire data in more homogeneous environment and all other applications are shut down to avoid any possible interruption.

Similarly, in remote execution mode, USB cable is disconnected and display brightness is set to 50%. The battery level is also maintained between 60 to 70% and in order to avoid fluctuations caused by excess heat in device, we stop data collection on regular intervals. The wireless network was isolated to avoid inference and fluctuations caused by other wireless network consumers.

## 5.2 Statistical Modeling

In this section, we describe our statistical analysis model in two parts; one for the application execution time and another for the energy consumed. Using the results of the statistical model we can verify reliability of the concluded results from the benchmarking experiments.

The RMA in this analysis is a service-based application consists of three mathematical utility services, namely prime, matrix multiplication, and matrix covariance. Prime and matrix operations are increasingly being employed in computational tasks including image processing, voice recognition, enterprise applications, navigation and graphical applications. They are also part of learning applications to help learners better grasp the mathematical foundations on their mobile devices.

The analysis is undertaken for two execution modes of local and PMC. The local execution mode necessitates local execution of the entire RMA including all the three services. However, in the PMC execution mode, the RMA execution is initiated inside the mobile service consumer device and the computational services are asynchronously called for execution outside the host device on three nearby resource-rich mobile devices. The general assumption in our proposal is that the MSC is a resource-constraint mobile device, the MSPs are resource-rich mobile devices in vicinity, and the TSG is deployed on MNO infrastructures.

For producing the statistical model of time and energy in local and PMC mode, we employ independent replication method to generate independent dataset consists of the measured execution time and consumed energy data for new independent workloads (there is no correlation between these workloads and those measured in benchmarking experiment) in local and PMC modes. The same testbed described in Section 5.1 is used. Using data in the independent dataset, we train linear regression model using IBM SPSS 22 to

identify the correlations between the workloads and execution time as well as between execution time and consumed energy to derive the regression models of time and energy. Using the regression models, the application execution time and consumed energy data can be generated to validate the performance evaluation findings undertaken via benchmarking analysis.

In order to validate our regression model, we leverage split-sample approach and perform calibration-validation exercise. Thus partial dataset is used to build the model and the rest to validate the results of the model. In order to perform validation, we randomly split the sample into two different size samples and identify the correlations between the dependent and independent variables. If the results are supporting each other, the model is valid.

In the following, we present an in-depth explanation of statistical model of execution time and consumed energy.

### 5.2.1 Execution Time

In formulating application execution time, we measure execution time of the application when the application is locally running on the mobile device and when it uses remote PMCs. The observed times which are called measured execution times are used to perform and train linear regression models so the statistical correlation is derived from the real data. For validating the derived execution time models, we use the second split of the dataset consists of workloads and corresponding execution times, and compare the result of measured execution time with the predicted execution time out of the regression model.

The input value for the linear regression model depends on the complexity of the given algorithm (application) which is calculated using BIG O notation $O(f(n))$, because accurate execution time estimation is usually impossible due to the strong dependency of

execution time and energy of application to the performance of underlying machine and the compiler by which the machine code is generated.

We describe the statistical modeling of time and energy in local and PMC execution modes as follows. For each execution mode, we describe time and energy separately. The devised models are validated using split-sample approach.

### *5.2.1 (a) Local Execution*

The three utility services in this model are locally executed on mobile device. Hence, the $TL_{RMA(i)}$ as the total execution time for the $i^{th}$ workload is formulated as:

$$TL_{RMA(i)} = T_{Prime(p_i)} + T_{Multiply(m_i)} + T_{Covariance(c_i)} \tag{5.1}$$

where $T_{Prime(p_i)}$, $T_{Multiply(m_i)}$, and $T_{Covariance(c_i)}$ are the maximum times to run the prime, multiply, and covariance services for the $i^{th}$ workload of $p$, $m$, and $c$ respectively. In the following, we describe the process of devising time models of prime, matrix multiply, and matrix covariance that help us build the main model.

- **Prime:** The first service executing in this RMA is the primality test to verify that the given workload is prime. The given workloads must be prime to perform computation, otherwise, if the given workload is not prime, there will not be much computation and the algorithm quickly responds without intensive computations. The Big O notation of running time for this service is polynomial and is linearly proportionate to the workload values. Hence, the upper execution time bound is $O(p)$.

  Therefore,

  $$T_{Prime(p_i)} = O(p_i) \tag{5.2}$$

  The complexity of the algorithm is used to identify the correlation between the work-

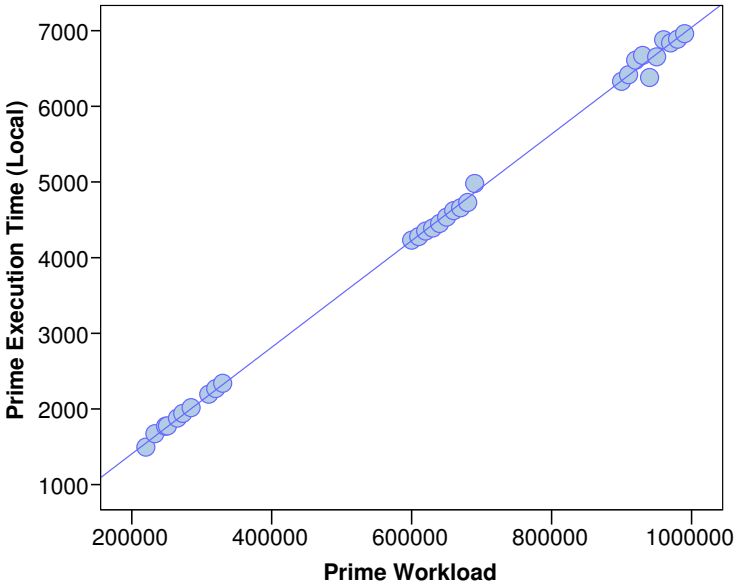



Figure 5.2: Linearity Correlation of Prime Workload and Execution Time in Local Execution Mode

load value and the measured execution time. Prior the linear regression modeling, the linearity of the Prime function needs to be ensured, otherwise the linear regression can be misleading. To demonstrate the linearity of correlations between workloads and time in prime function, we depicted the scatter diagram that is depicted in Figure 5.2. As the results show, the relationship between the workload values and respective execution times of Prime function is linear, as expected. Thus, the linear regression is feasible to model this relationship.

For the prime task, the accuracy of the statistical model depends on the $\beta$ and $\alpha$ in the following equation which is determined using linear regression. The $\beta$ is the coefficient of the prime workload and $\alpha$ is the constant value depending on the execution device and runtime environment. Hence, the prime execution time model is

$$T_{Prime(p_i)} = (\beta \times p_i) + \alpha \tag{5.3}$$

where $p_i$ is the independent variable in our model and is the $i^{th}$ workload value. For instance, $T_{Prime(1000)} = \beta \times 1000 + \alpha$, where $\beta$ is the coefficient and $\alpha$ is the

constant value. The execution time for the selected workloads are measured while executing on the mobile device. In order to determine the $\alpha$ and $\beta$ values, we utilize workloads of the measured dataset and their corresponding execution time to train the linear regression model. The results of the linear regression analysis for the prime are given in Table 5.2. The $R$ value in the Table testifies very strong correlation between the workload value and its execution time. The $R^2$ in the Table explains that the execution time values can be 99.9% explained using the given workloads. Adjusted $R^2$ ensures that the predicator (workload values as independent variable) is an appropriate regressor.

Table 5.2: Linear Regression Model Summary for Prime Application in Local Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig. |
|---|---|---|---|---|---|
| 1 | 0.999804 | 0.999609 | 0.999579 | 33227.69 | 0.000 |

The $F$ and $Sig$ values in the Table show significant direct correlations between the workload and execution time. These results enable us to leverage linear regression to derive our statistical model for the execution time of the prime application.

The linear regression analysis determines the unstandardized coefficients values for the execution time of the prime as $\beta = 0.007$ and $\alpha = 34.512$. Hence, the statistical model for the execution time of the prime application in local mode is:

$$T_{Prime(p_i)} = (0.007 \times p_i) + 34.512 \tag{5.4}$$

This Equation is used to generate time data that will be presented in next chapter.

- **Matrix Multiplication:** For the matrix multiplication, the service receives two $[T * S][S * K]$ matrices and calculates the product of two matrices to produce the result as a $[T * K]$ matrix. The multiplication algorithm is implemented using three loops

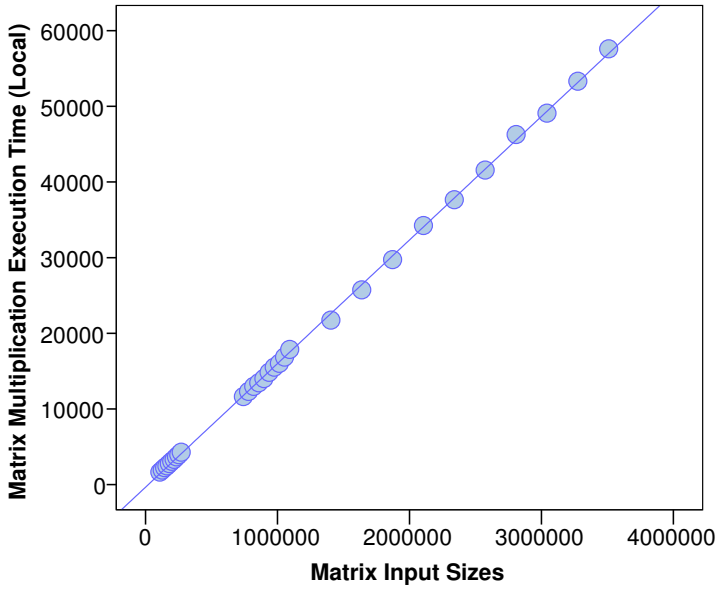



Figure 5.3: Linearity Correlation of Matrix Multiplication Workload and Execution Time in Local Execution Mode

with running time of $O(m^3)$ that $m$ represents the dimension of the given matrix as workload. Therefore, the upper runtime bound for the matrix multiplication service is

$$T_{Multiply(m_i)} = O(T_i \times S_i \times K_i) \tag{5.5}$$

Similar to the prime, the multiplication complexity is used an input variable to the regression model. For the sake of clarity and accuracy and to avoid complex calculations, we consider $m_i = T_i \times S_i \times K_i$. Hence, the statistical model of execution time for the matrix multiplication is:

$$T_{Multiply(m_i)} = (\beta \times m_i) + \alpha \tag{5.6}$$

which is a linear equation.

Before performing regression modeling, the linearity of matrix multiplication function must be ensured; otherwise the linear regression can be misleading. The scatter diagram for matrix multiplication workloads and corresponding execution times is

appeared in Figure 5.3. As the results show, the relationship between the workload sizes and respective execution time of matrix multiplication function as expected is linear. Thus the linear regression is feasible to model this relationship.

The linear regression results for determining the $\alpha$ and $\beta$ are summarized in Table 5.3.

Table 5.3: Linear Regression Model Summary for Matrix Multiply Application in Local Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig. |
|---|---|---|---|---|---|
| 1 | 1.000 | 1.000 | 1.000 | 145737.5 | 0.000 |

In order to determine the $\alpha$ and $\beta$ values, we utilize workload sizes of the measured dataset and their corresponding execution time to train the linear regression model. The results of the linear regression analysis for the matrix multiplication are given in Table 5.3. The $R$ value in the Table testifies full correlation between the workload value and its execution time. The $R^2$ in the Table explains that the execution time values can be 100% explained using the given workload dimension. Adjusted $R^2$ ensures that the predicator (workload sizes as independent variable) is an appropriate regressor. The results of $\beta$ and $\alpha$ coefficients are determined by the linear regression as $-43.560$ and 0.016 respectively. Therefore, the execution time model of the covariance application in local execution mode is

$$T_{Multiply(m_i)} = (0.016 \times m_i) - 43.560 \tag{5.7}$$

- **Covariance:** Similarly, the covariance algorithm has its own runtime and complexity. The covariance service, receives a single square $[N * N]$ matrix and produces its covariance matrix of the same size. Similar to the multiplication service, the upper

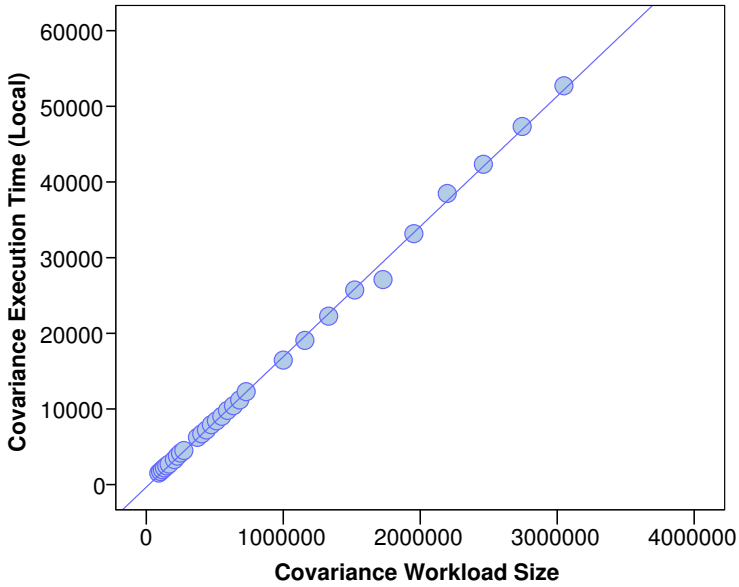



Figure 5.4: Linearity Correlation of Covariance Workload Sizes and Execution Time in Local Execution Mode

runtime bound of the covariance service is $O(N^3)$. Therefore,

$$T_{Covariance(c_i)} = O(N_i \times N_i \times N_i) \tag{5.8}$$

Similar to the matrix multiplication, we consider $C_i = N_i \times N_i \times N_i$. The complexity is used to perform the linear regression. Therefore, the statistical model of execution time for the matrix covariance is

$$T_{Covariance(c_i)} = (\beta \times C_i) + \alpha \tag{5.9}$$

Before performing regression modeling, the linearity of matrix multiplication function must be ensured; otherwise the linear regression can be inappropriate. The scatter diagram for matrix multiplication workloads and corresponding execution times is presented in Figure 5.4.

As the results show, the relationship between the workload values and respective execution time of Prime function as expected is linear. Thus the linear regression is

feasible to model this relationship. The linear regression results to determine the $\beta$ and $\alpha$ are presented in Table 5.4.

Table 5.4: Linear Regression Model Summary for Covariance Application in Local Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig. |
|---|---|---|---|---|---|
| 1 | 1.000 | 1.000 | 1.000 | 36112.8 | 0.000 |

The $R$ value in Table 5.4 shows significant correlation between the workload and its execution time. The $R^2$ value indicates that the application execution time entirely depends on its workload values and complexities. Adjusted $R^2$ ensures that the predicator (matrix dimension of workloads as independent variable) is an appropriate regressor.

The $F$ and $Sig$ values show that available data fulfill assumptions of linear regression and therefore, we are able to perform linear regression. The $\alpha$ and $\beta$ values of the linear regression for execution time of the covariance application are $-1.193$ and 0.0017, respectively. Therefore, the statistical model of covariance application execution time in local execution mode is:

$$T_{Covariance(c_i)} = (0.017 \times C_i) - 1.193 \tag{5.10}$$

The application execution time results driven from this Equation will be presented in next chapter.

For calculating the entire application execution time in local execution mode, we can substitute the run time of each application into Equation (5.1), as following:

$$TL_{RMA(i)} = (0.007 \times p_i + 34.512) + (0.016 \times m_i - 43.560) + (0.017 \times c_i - 1.193) \tag{5.11}$$

The Equation (5.11) is our statistical model by which total application execution time is calculated for each set of workloads (i.e., p, m, c). The generated results using this Equation are presented in next chapter.

To validate our devised model, we used split-sample procedure. Using random function in SPSS 22, we split our sample into two randomly selected partitions. For each partition, we determine the correlation coefficients and compare the results of both partitions with the unpartitioned sample to ensure validity. The results of our comparison are presented in Table 5.5. As the results show, the model produces identical R, $R^2$, and adjuster $R^2$ for all three different samples. The degree of freedom (df) column in the table shows that the size of each sample is unique and random. The adjusted $R^2$ is similar for all the groups which is an evidence on the validity of our proposed statistical model of local execution time.

Table 5.5: Comparison of Split-Sample Approach Results for Validation of Local Execution Time Model

| Metrics | split = 1.00 (Selected) | split = 0.00 (Selected) | Non-split Sample |
|---|---|---|---|
| R | 1 | 1 | 1 |
| $R^2$ | 1 | 1 | 1 |
| Adjusted $R^2$ | 1 | 1 | 1 |
| df | 18 | 10 | 29 |

### *5.2.1 (b) PMC Execution*

PMC execution, as the second execution mode, is when the local execution of the RMA reaches an intensive service that should be called for remote execution in proximate mobile devices. The application calls the TSG, forwards the service names, and receives the IP addresses of nearby mobile devices able to perform desired computations (if found). The IP addresses are used by the mobile client to build the IP addresses and asynchronously call execution of services using the given IP address. Upon successful execution of the services by the remote MSPs, the results are sent back and integrated with

the local code. In PMC execution mode, the entire intensive executions take place outside the mobile device and consequently conserves noticeable amount of mobile battery.

$TR_{RMA(i)}$ is the maximum time that the entire cycle of the $i^{th}$ workload takes for execution and is:

$$TR_{RMA(i)} = T_{Discovery} + TW_{Remote-Computing(RMA(i))} + T_{Communication(RMA(i))} \quad (5.12)$$

where $T_{Discovery}$ is the communication and computation time that the TSG takes to identify nearby mobile service providers capable of executing required services. $T_{Discovery}$ value is regardless of the workload values and types. Since computation time of discovering MSPs depends on performance of the host server and database engine in TSG, its value is negligible considering high performance of contemporary servers and databases. Hence, since communication overhead is likely more important in our framework, we consider the communication and computation time as a single accumulated delay and consider its impact in Equation (5.13) where we formulate the total communication delay.

$TW_{Remote-Computing(RMA(i)}$ is the maximum waiting time required for the remote server to execute the $i^{th}$ RMA which is defined as follows:

$$TW_{Remote-Computing(RMA(i))} = T_{Discovery} + TW_{Prime(p_i)} + TW_{Multiply(m_i)} + TW_{Covariance(c_i)} \quad (5.13)$$

where $TW_{Prime(p_i)}$, $TW_{Multiply(m_i)}$, and $TW_{Covariance(c_i)}$ are waiting time for PMC execution of the $i^{th}$ prime, multiply, and covariance workload inside the remote servers. However, since the tasks are asynchronously executed on three different MSP, the total time is

$$TW_{Remote-Computin(RMA(i))} = T_{Discovery} + Max(TW_{Prime(p_i)}, TW_{Multiply(m_i)}, TW_{Covariance(c_i)}) \quad (5.14)$$

Considering the workload intensities and size in Table 5.1, the most complex computing task is the matrix multiplication. Hence, the Equation (5.13) is

$$TW_{Remote-Computing(RMA(i))} = TW_{Multiply(m_i)} \tag{5.15}$$

where we omit the $T_{Discovery}$ because it is considered while modeling the communication latency later in this section.

Unlike execution time in local execution mode that depends on the computing power of the host mobile device, in PMC execution mode, the computing time strongly depends on the computing capabilities of the remote servers which are not identical even if the computing specifications are the same (due to existing heterogeneity). The statistical model of PMC execution time for the multiply task, depends on the $\beta$ and $\alpha$ in Equation (5.16). In order to accurately estimate the $\beta$ and $\alpha$ values, the measured waiting time of PMC execution of multiply task are used to draw linear regression. Inside the remote server, the complexity of the multiply algorithm defines the execution time. Hence, the input to the linear regression is the complexity of the multiply algorithm and their workloads. As described in local execution mode, the complexity of multiply application is $O(m^3)$. So the size of the matrices that are sent for PMC execution, called $m_i$, are used to measure the application complexity.

$$TW_{Multiply(m_i)} = (\beta \times m_i) + \alpha \tag{5.16}$$

Before we perform regression modeling, we depict the linearity of matrix multiplication and PMC execution time. We have depicted the scatter diagram for matrix sizes of workloads and corresponding execution times in Figure 5.5.

As the results show, the relationship between the workload sizes and respective execution time is linear. Thus the linear regression is feasible to model this relationship.

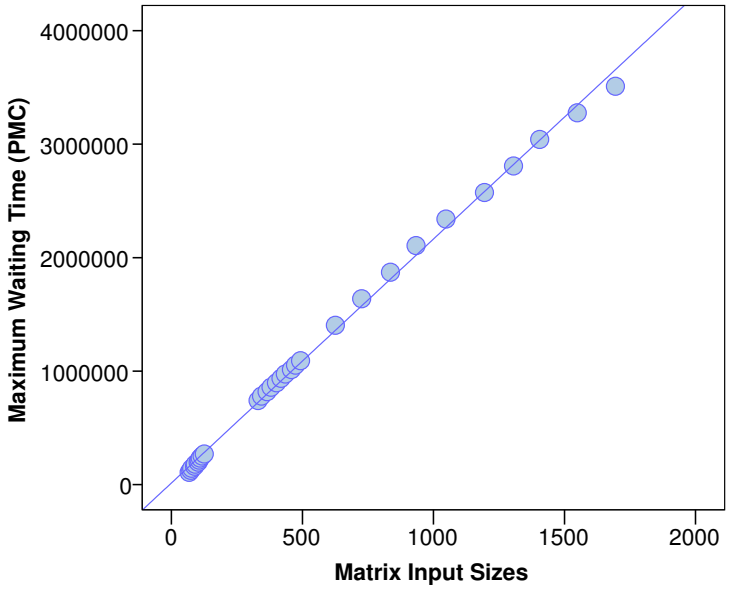



Figure 5.5: Linearity Correlation of Matrix Multiplications Workload Size and Execution Time in PMC Execution Mode

Table 5.6: Linear Regression Wait Time Model Summary for Multiply Application in PMC Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig. |
|---|---|---|---|---|---|
| 1 | 1.000 | 0.999 | 0.999 | 16253.67 | 0.000 |

The results of the linear regression are given in Table 5.6. The $R$ value in Table testifies %100 correlation between the workload size value and its corresponding PMC execution waiting time. The $R^2$ explains that %99.9 of the waiting time values can be explained using the given workload sizes.

The $F$ and $Sig$ values in the Table show significant direct correlations between the workload size and the waiting time for PMC execution. These results enable us to leverage linear regression to derive our statistical model for the waiting time of the multiply application. The linear regression analysis determines the coefficients values for the waiting time of the multiply as $\beta = 0.000425$ and $\alpha = 15.785$.

Hence, the statistical model of waiting time for the PMC execution time is written as:

$$TW_{Multiply(m_i)} = (0.000425 \times m_i) + 15.785 \tag{5.17}$$

by replacing the right side of Equation (5.17) into Equation (5.15), we have

$$TW_{Remote-Computing(RMA(i))} = (0.000425 \times m_i) + 15.785 \quad (5.18)$$

In our proposed model, four remote executions calls to out PMC resources are performed. One is when the MSC sends discovery request to the TSG to obtain the IP addresses of the nearby MSPs and three remote execution calls are performed when the MSC sends three individual requests to the MSPs for computing the service. Therefore, $T_{Communication(i)}$ is

$$T_{Communication(i)} = T_{Discovery} + TC_{Prime(p_i)} + TC_{Multiply(m_i)} + TC_{Covariance(c_i)} \quad (5.19)$$

where $T_{Discovery}$ is the total time of discovering IP addresses of MSP from the TSG. $TC_{Prime(p_i)}, TC_{Multiply(m_i)}$, and $TC_{Covariance(c_i)}$ are communication time to transfer the request and response between the MSC and respective MSPs. However, similar to 5.14, since the tasks are asynchronously called, the total communication time of executing three services, is the time that the longest service take for completion. Therefore, we have

$$T_{Communication(i)} = T_{Discovery} + MAX(TC_{Prime(p_i)}, TC_{Multiply(m_i)}, TC_{Covariance(c_i)}) \quad (5.20)$$

The communication time for each task is being formulated in chapter 3 and is appeared as Equation (3.7). So, for each communication, the Equation (3.7) stands valid and applicable. However, both factor of number of hops and distance between MSC and MSP are fixed for Prime, Multiply, and Covariance and the only factor that changes among these three is the data volume. The more data is transmitted the more is the communication overhead. Considering communication volume of each workload presented in Table 5.1, the most data-intensive component is multiply. Hence, we the communication overhead is

formulated as follows:

$$T_{Communication(i)} = T_{Discovery} + TC_{Multiply(m_i)} \tag{5.21}$$

However, because Transmission Control Protocol (TCP) communication is used between MSC and MSP/TSG, the real time delays and constraints of the TCP communications such as Maximum Transmission Unit (MTU), Congestion Window (CWnd), and slow-state threshold impose significant impacts on the communication time and complicate its modeling, especially when the communication volume increases. Therefore, in order to produce an accurate statistical model for the time, we use linear regression which helps us to derive more accurate estimation function for the delay time of the PMC execution.

The linear regression analysis is needed to determine the $\alpha$ and $\beta$ for the Equation (5.22). Before performing regression modeling, the linearity of communication delay should be ensured otherwise the linear regression can be inappropriate. The scatter diagram for communication delay and corresponding independent values (discovery delay and $TC_{Multiply}$ is plotted in Figure 5.6. As the results of the 3-D plot show, the relationship is linear as expected. Thus the linear regression is feasible to model this relationship.

$$T_{Communication(i)} = (\gamma \times T_{Discovery}) + (\beta \times TC_{Multiply(m_i)}) + \alpha \tag{5.22}$$

Table 5.7: Linear Regression Model Summary for Communication Delay in PMC Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Simga |
|---|---|---|---|---|---|
| 1 | 0.995 | 0.990 | 0.988 | 577.593 | 0.000 |

The results of the linear regression are given in Tables 5.7. The $R$ value in this Table testifies %99.5 correlation between the communication delay and two independent

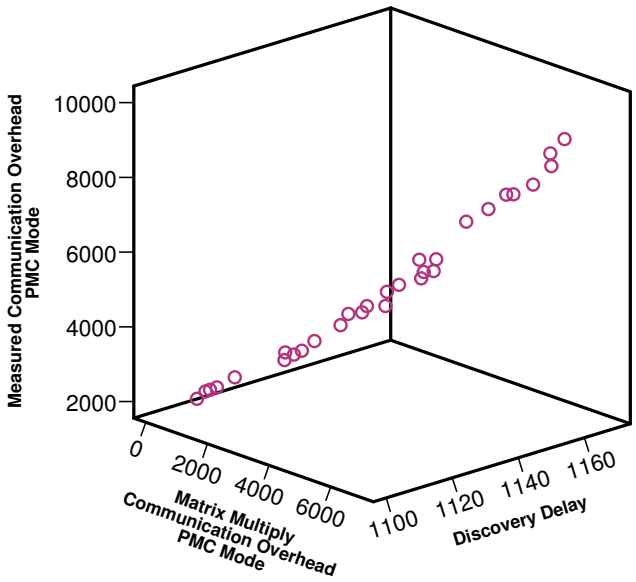



Figure 5.6: Linearity Correlation of Communication Delay in PMC Execution Mode

variables of discovery delay and communication overhead of multiply component. The $R^2$ explains that %99 of the communication overhead can be explained using the given variables.

The $F$ and $Sig$ values show significant direct correlations between the communication overhead and the variables. These results enable us to leverage linear regression to derive our statistical model for the communication overhead in PMC execution mode. The linear regression analysis determines the coefficients values for the waiting time of the multiply as $\beta = 0.98, \gamma = 1.30$ and $\alpha = -47.28$. Hence, we have

$$T_{Communication(i)} = (1.3 \times T_{Discovery}) + (0.98 \times TC_{Multiply(m_i)}) - 47.28 \tag{5.23}$$

Hence, considering Equations 5.18 and 5.23 the statistical model for the total execu-

tion time of the $RMA(i)$ in PMC execution is

$$TR_{RMA(i)} = (0.0000425 \times m_i) + (1.3 \times T_{Discovery(i)}) + (0.98 \times TC_{Multiply(m_i)}) + (15.785 - 47.28) \quad (5.24)$$

To validate our devised model, we perform split-sample procedure explained earlier. We split the sample data into two randomly selected partitions using the randomly generated values for local execution time. For each partition, we determine the correlation coefficients and compare the results of both partitions with the full sample to ensure validity. The comparison results are summarized in Table 5.8. As the results show, the model produces identical R, $R^2$, and adjuster $R^2$ for all three different samples. The degree of freedom (df) column in the table shows that the size of each sample is unique and random. The adjusted $R^2$ is alike for all the splits which is an evidence on the validity of our proposed statistical model

Table 5.8: Comparison of Split-Sample Approach Results for Validation of PMC Execution Time Model

| Metrics | split = 1.00 (Selected) | split = 0.00 (Selected) | Non-split Sample |
|---|---|---|---|
| R | 1 | 1 | 1 |
| $R^2$ | 1 | 1 | 1 |
| Adjusted $R^2$ | 1 | 1 | 1 |
| df | 18 | 10 | 29 |

We produce data of execution time for local and PMC execution modes using Equations 5.11 and 5.24 respectively. These data are presented in next chapter.

### 5.2.2 Consumed Energy

In this section, we present the statistical model to produce energy data in local and PMC execution modes. First part, introduces the statistical model to formulate the energy consumed in local execution mode followed by second part for the PMC execution mode.

Similar to the validation approach used in execution time, we validate our energy model of local and PMC.

### *5.2.2 (a) Local Execution*

Energy consumption of the mobile device running a RMA mainly comprises of the total energy used by the CPU and Liquid Crystal Display (LCD). However, we disregard energy consumption of LCD since it has no dependency to the processing requirements of the mobile device. Moreover, in local execution mode, there will not be any wireless communication. Hence, the only power consuming components is CPU. If $EL_{RMA(i)}$ is the total energy consumed for the local execution of the $i^{th}$ workload, therefore we have

$$EL_{RMA(i)} = EL(CPU_i) \tag{5.25}$$

where $EL(CPU_i)$ is the CPU energy consumed to locally execute the entire $i^{th}$ workload.

CPU power consumption for each computational component highly depends on the execution time of that particular component, and execution time itself is a direct functional of the workloads. So, the intenser is the workload, the higher will be the execution time, and the more will be CPU power consumption.

Because of significant dependency of the consumed energy to the execution time, we study the consumed energy of the RMA as whole regardless of the energy consumption of each component. Therefore, we consider the RMA in whole for presenting energy model of our model.

Similar to statistical model of execution time, in order to present a reliable and accurate estimation model of the CPU energy, we perform linear regression using measured real data on the mobile device. We use datasets of workloads including the execution time and energy consumption of each workload and use them for training the regression model

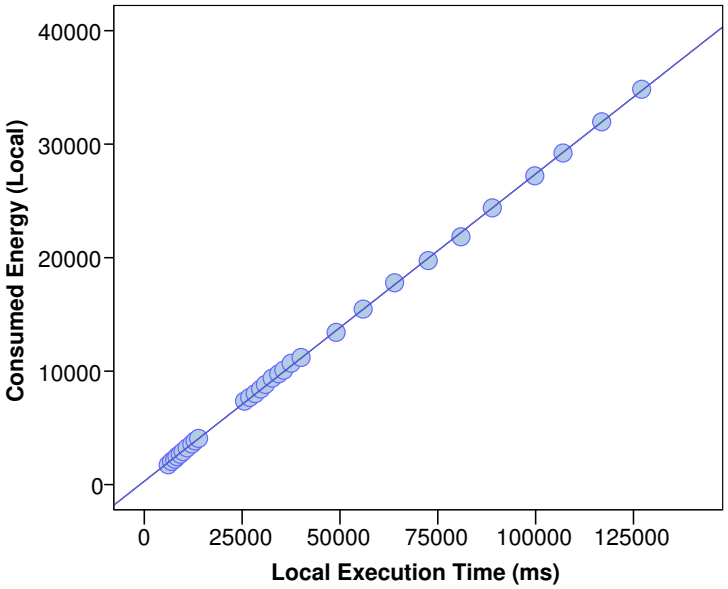



Figure 5.7: Linearity Correlation of Consumed Energy and Time in Local Execution Mode

using SPSS 22 to produce the energy model. For validation of our proposed model, we use the split-sample approach.

The mobile device used in this modeling is HTC Nexus One that features a Reduced Instruction Set Computing (RISC) 32-bit Qualcomm Snapdragon S1 QSD8250 chipset[4]. This chip has a single core ARMv7 application processor with maximum 1024 MHz clock frequency. This CPU consumes $\beta$ milli watts power per second, where $\beta$ is the power consumption coefficient for this particular processor at $1024MHz$ clock frequency which can be determined using linear regression. Hence, the power model can be presented as a function of execution time written as:

$$EL(CPU_i) = (\beta \times TL_{RMA(i)}) + \alpha \tag{5.26}$$

where $TL_{RMA(i)}$ is the total execution time for the $i^{th}$ workload and $\alpha$ is a constant values. Both $\beta$ and $\alpha$ values can be determined using training over linear regression.

Prior to regression analysis, the type of regression should be identified whether it

[4]http://www.qualcomm.com/media/releases/2007/11/14/qualcomm-premieres-snapdragon-first-chipset-solutions-break-gigahertz

is linear or non-linear. The scatter diagram for the local energy consumption is plotted in Figure 5.7. As the results show, the relationship between the energy and respective execution time of workloads is linear. Thus, we perform linear regression to model this relationship and derive the statistical model.

The detail statistics of the statistical model of our linear regression are summarized in Table 5.9. The $R$ value shows significant correlation between the execution time and consumed energy. The $R^2$ value in the Table testifies that 100% of the energy values can be explained using execution time due to significant direct correlation. Values presented in adjusted $R^2$ column advocates that the predicator (time) is an appropriate regressor to model energy. The F and sigma values in the Table ensure that available dataset is appropriate to be used for linear regression. Hence, performing linear regression is applicable to our model and derived model is reliable and beneficial.

Table 5.9: Mathematical Model Summary of the Consumed Energy in Local Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig. |
|---|---|---|---|---|---|
| 1 | 1.000 | 1.000 | 1.000 | 36396.4 | 0.000 |

The coefficient values for the regression model are $\beta = 0.281$ and $\alpha = 119.587$. Hence, Equation (5.26) is :

$$EL(CPU_i) = (0.281 \times TL_{RMA(i)}) + 119.587 \tag{5.27}$$

Equation (5.27) is used to produce energy data for local execution mode. The execution time values extracted from Equation (5.11) are used as input to this Equation. The result of data generation appeared in next chapter.

Validation of the devised energy model of local execution mode is carried out using the split-sample procedure. The sample is divided into two randomly selected partitions.

For each partition, we determine the correlation coefficients and compare the results of both partitions with the full sample to demonstrate validity. The results of our comparison are presented in Table 5.10. As the results show, the model produces identical R, $R^2$, and adjuster $R^2$ for all three different samples. The degree of freedom (df) column in the table shows that the size of each sample is uniquely accidental. The adjusted $R^2$ is identical for all the splits which is an evidence on the validity of our proposed statistical model.

Table 5.10: Comparison of Split-Sample Approach Results for Validation of Local Energy Consumption Model

| Metrics | split = 1.00 (Selected) | split = 0.00 (Selected) | Non-split Sample |
|---|---|---|---|
| R | 1 | 1 | 1 |
| $R^2$ | 1 | 1 | 1 |
| Adjusted $R^2$ | 1 | 1 | 1 |
| df | 18 | 10 | 29 |

### *5.2.2 (b) PMC Execution*

Similar to the local execution, we disregard energy consumption of the LCD for PMC execution. However, in PMC execution CPU and Wi-Fi are two major energy consumers which will be considered for devising the energy model. In PMC execution mode, the resource discovery service is consuming some amount of energy. Hence, if $ER_{RMA(i)}$ is the total energy consumed for PMC execution of the $i^{th}$ workload, therefore we have

$$ER(RMA_{(i)}) = ER(Discovery) + ER(TW_{Remote-Computing(i)}) + ER(WiFi_{(i)}) \qquad (5.28)$$

where $ER(Discovery)$ is local energy consumption to perform remote MSP discovery, including the energy consumed for waiting for the results from TSG. $ER(TW_{Remote-Computing(i)})$ is the total energy consumed while device is waiting for the results to come from the remote server. $ER(WiFi_{(i)})$ is the total energy consumed during all communication with external entities, including TSG and MSP(s) for $i^{th}$ workload.

Hence, the statistical model of the consumed energy in PMC execution model depends on three components of

- $TW_{Remote-Computing(i)}$: execution time of the maximum component (Matrix Multiply in this experiment)
- *Discovery*: discovery time
- $D_i$: total data volume transmitted between MSC and MSP in each workload

For any component, except discovery delay (which is almost constant regardless of workload value), the linear regression can be performed to produce a statistical model. Discovery delay has no coefficient such as workload size, data volume and so on. It only depends on the wireless communication quality between the MSC and TSG.

However, before we can perform regression analysis, the type of relationship between the dependent and independent variable should be identified; if it is linear or not. The scatter diagram for the remote energy consumption is plotted in Figure 5.8. As the results show, the relationship between the energy and respective execution time of workloads

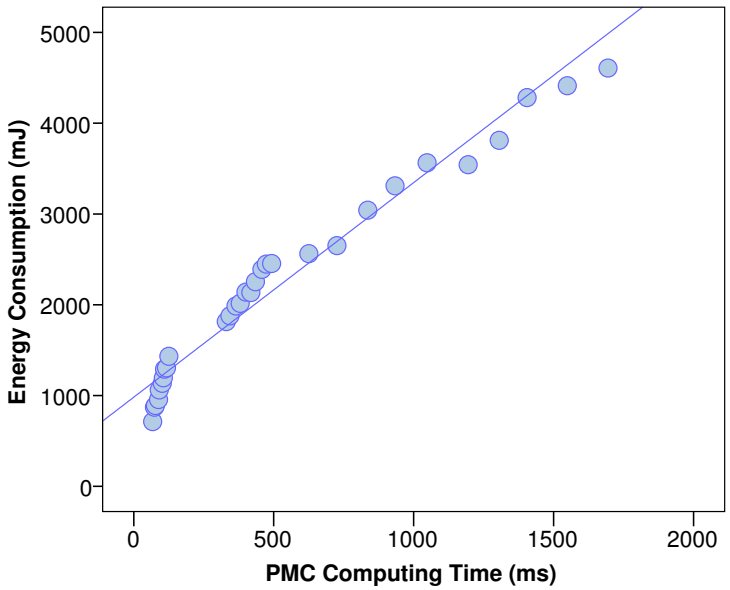



Figure 5.8: Linearity Correlation of Consumed Energy and PMC Computing Time in PMC Execution Mode

is linear. Thus, we perform linear regression to model this relationship and derive the statistical model and identify the $\alpha$ and $\beta$.

For the $TW_{Remote-Computing(i)}$, the possible statistical model is:

$$ER(TW_{Remote-Computing(i)}) = (\beta \times TW_{Remote-Computing(i)}) + \alpha \tag{5.29}$$

Table 5.11 presents the results of statistical model trained in SPSS for $TW_{Remote-Computing(i)}$.

Table 5.11: Mathematical Model Summary of the CPU Consumed Energy in PMC Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig. |
|---|---|---|---|---|---|
| 1 | 0.960 | 0.921 | 0.92 | 151.817 | 0.000 |

As $R$ value shows in Table 5.11 shows, the given metrics have %96 correlation with the $TW_{Remote-Computing(i)}$ energy consumption in PMC execution mode. The $R^2$ explains that %92 of the total energy can be explained using $TW_{Remote-Computing(i)}$ in this regression. The $F-test$ results and $Sig$ value in the Table shows significant direct correlations between the $TW_{Remote-Computing(i)}$ and the total energy consumed for remote computation of the task. These results permits to leverage linear regression to derive the statistical model. The linear regression analysis determines the coefficients values for the waiting time of the multiply as $\alpha = 746.373$, $\beta = 3.415$. Hence, we have

$$ER(TW_{Remote-Computing(i)}) = (3.41 \times TW_{Remote-Computing(i)}) + 746.373 \tag{5.30}$$

Similarly the model can be produced for the energy consumed by the $Wi-Fi$. We draw the scatter diagram to show the linearity of energy consumed by Wi-Fi and the data transfer volume. The scatter diagram for the remote Wi-Fi energy consumption is illus-

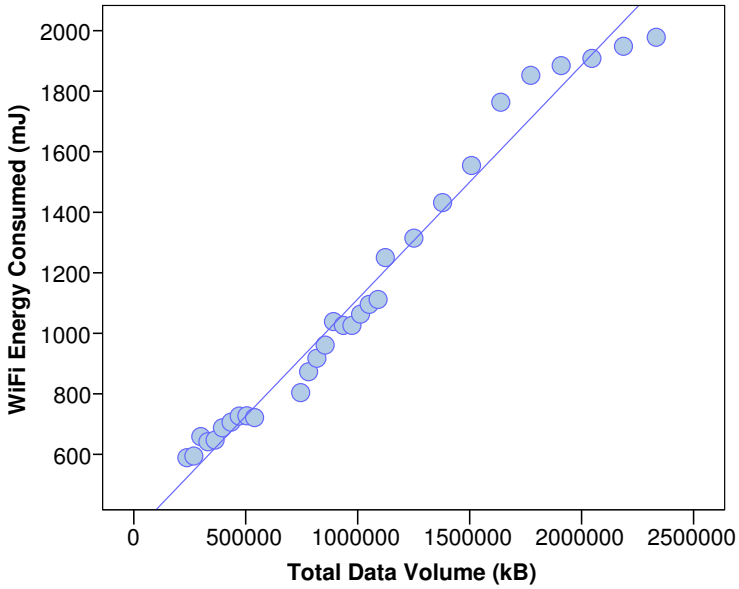



Figure 5.9: Linearity Correlation of Wi-Fi Consumed Energy and Data Volume in PMC Execution Mode

trated in Figure 5.9.

As the results show, the relationship between the energy and respective data volume of workloads is linear. Thus, we perform linear regression to model this relationship and derive the statistical model and identify the $\alpha$ and $\beta$ for Wi-Fi energy.

$$ER(WiFi_{(i)}) = (\beta \times D_i) + \alpha \tag{5.31}$$

Table 5.12 presents the results of statistical model trained in SPSS for $WiFi_{(i)}$.

Table 5.12: Mathematical Model Summary of the Wi-Fi Consumed Energy in PMC Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig. |
|---|---|---|---|---|---|
| 1 | 0.970 | 0.940 | 0.94 | 203.5 | 0.000 |

As $R$ value shows in Table 5.12 shows, the given metrics have %97 correlation with the $Wi-Fi$ energy consumption in PMC execution mode. The $R^2$ explains that %94 of the total energy can be explained using $Wi-Fi$ in this regression. The $F-test$ results and $Sig$ value in the Table shows significant direct correlations between the $TW_{Remote-Computing(i)}$ and the total energy consumed for remote computation of the task. These results permits

to leverage linear regression to derive the statistical model. The linear regression analysis determines the coefficients values for the waiting time of the multiply as $\alpha = 446.468274$, $\beta = 0.000579$. Hence, we have

$$ER(TW_{WiFi_{(i)}}) = (0.000579 \times D_i) + 446.468274 \tag{5.32}$$

Therefore, the total energy model for the PMC execution mode is

$$ER(RMA_{(i)}) = (\alpha \times Discovery_{(i)}) + (\beta \times TW_{Remote-Computing(i)}) + (\omega \times WiFi_{(i)}) + \theta \tag{5.33}$$

where $TR_{RMA(i)}$ is the total remote execution time for the $i^{th}$ workload. $\alpha, \beta, \omega$ are respective coefficients and $\theta$ is the constant value that will be determined via linear regression.

The detail statistics of the statistical model of our linear regression for the PMC execution mode are summarized in Table 5.13. The $R$ value shows significant correlation between the execution time and consumed energy. The $R^2$ values in the Table testifies that 99.5% of the energy values can be explained using given metrics due to significant correlations.

Table 5.13: Mathematical Model Summary of the Consumed Energy in PMC Execution Mode

| Model | R | $R^2$ | Adjusted $R^2$ | F | Sig |
|---|---|---|---|---|---|
| 1 | 0.998 | 0.995 | 0.99 | 756.17 | 0.000 |

The $F$ and *sigma* values in the Table ensure that available dataset is appropriate to be used for linear regression. Hence, performing linear regression is applicable to our model and derived model is reliable and beneficial. The coefficient values for the regression model are $\alpha = 3.117$, $\beta = 0.0938$, $\omega = 0.002$, and $\theta = -2666.364$. Hence, Equation

(5.34) is rewritten as :

$$ER(RMA_{(i)}) = (3.117 \times Discovery_{(i)}) + (0.0938 \times TW_{Remote-Computing(i)}) - (0.002 \times WiFi_{(i)}) - 2666.364 \quad (5.34)$$

Equation 5.34 is used to produce energy data for PMC execution mode. The execution time values extracted from Equation (5.24) are used as input to this Equation. The result of data generation appeared in next chapter.

To validate our devised PMC energy model, we employ the split-sample procedure. For this purpose, we divide the sample dataset into two randomly selected partitions. For each partition, the correlation coefficients are identified and the results of both partitions are compared with the unpartitioned sample to ensure validity. The results of our comparison are presented in Table 5.14. As the results indicate, the PMC energy model generates identical R, $R^2$, and adjuster $R^2$ for all three different samples. The degree of freedom (df) column in the table shows that the size of each sample is unique. The adjusted $R^2$ is identical for all the splits which is an evidence on the validity of our proposed statistical model.

Table 5.14: Comparison of Split-Sample Approach Results for Validation of PMC Energy Consumption Model

| Metrics | split = 1.00 (Selected) | split = 0.00 (Selected) | Non-split Sample |
|---|---|---|---|
| R | 1 | 1 | 1 |
| $R^2$ | 1 | 1 | 1 |
| Adjusted $R^2$ | 1 | 1 | 1 |
| df | 18 | 10 | 29 |

## 5.3 Platform-Independence Experiment

In this section, we perform another set of benchmarking experiments on a separate test-bed (separate from our evaluation test-bed) to demonstrate that the performance of

our proposed framework is not influenced by the programming language, mobile device, mobile operating system, and benchmarks that we have selected to performed the time-energy performance evaluation experiments.

### 5.3.1 Experiment Setup

**• Test-bed's Computing and Communicating Specification:**

We have build a separate test-bed consists of five computing devices; three Dell XPS14z laptops each of them featuring an Intel i5-2450M chipset with 2.5 GHz clock speed CPU, 4 GB RAM, 1 TB storage, running 64-bit Windows 7 Professional accessible via wireless access point that serve as MSPs in this test-bed. We have used another Dell Laptop Vostro 1200 featuring Intel Core 2 Duo T7250 CPU, 4 GB RAM, and 160 GB storage running windows 7 Ultimate that plays the role of MSC. We also leverage a Dell desktop that holds our SQL database and serves as our TSG. The desktop computer features Intel i5-2500 processor having 3.3 GHz clock speed, 4 GB memory, 1 TB storage, and 32-bit Windows 7 professional. The wireless access point used in this testbed is Cisco Linksys WRET54G running Tomato v1.28 as firmware. We have explicitly and intentionally selected our hardware devices different from what we had in our benchmarking experiments in Section 5.1 to highlight that the lightweight feature of our framework does not depend on the testbed and mobile device running as MSC and MSP.

**• Prototype Implementation:**

We implemented our framework and prototype using two different programming language; ASP and PHP. ASP (Active Server Page) is based on Microsoft whose codes are executed using Internet Information Service (IIS) web server on each mobile device. For executing PHP codes, we utilize Apache Tomcat web server. Change in programming language aims to demonstrate that performance of our framework does not depend on any particular

programming language.

• **Independent Benchmarks/Workloads:**

We have also used separate/independent benchmarks for this experiment aiming to show that any benchmark can be executed on our framework with approximately same performance gain. we employ Sieve of eratosthenes benchmark (Sieve in brief) which is an standard benchmark used for performance evaluation of computing systems (Jain, 2008). Sieve algorithm receives benchmark value $N$ and generates all the prime numbers from 1 to $N$.

Benchmark values are usually selected by experimenter as suggested by (Jain, 2008) stating that "in the Sieve benchmark the number of prime numbers to be generated can be set by the experimenter" (Jain, 2008). Thus, for selecting benchmark values $N$, we performed several pilot tests to determine appropriate boundaries for our benchmarks. It is known that the temporal cost of running benchmark $N$ must exceed the temporal cost of performing pre-offloading (i.e., MSP discovery in our model) in MCC (Kumar & Lu, 2010) to produce performance gain, otherwise the overhead surpass the performance gain.

As we explained in 4.2.1, our framework performs a resource discovery for any call that is identical for all benchmarks regardless of their values. This delay (including communication and computing delays) exists for any benchmark. Thus our benchmarks should be selected in a way that their computing time exceeds discovery delay. Therefore, we performed several pilot runs and could estimate this value as about 1000$ms$ for each call. This time is imposed as overhead on any benchmark used in our framework. According to research in (Kumar & Lu, 2010), the computing time of workloads to be offloaded is required to be more than the pre-offloading overhead which is about 1000$ms$ in our framework. The first $N$ value that fulfill this criteria is 9000. Hence, we choose our benchmark considering larger than 9000 and the maximum value of $N$ to be executed on our mobile

device as MSC is 27000. Therefore, our benchmarks are between 9000 to 27000 with equal step of 1800. Thus, the 10 benchmarks are $9000, 10800, 12600, ..., and 25200$. As suggested by (Jain, 2008), 10 benchmarks are sufficient to demonstrate execution performance of any computer system via Sieve benchmark. Thus, we selected 10 benchmarks in our experiment that are listed in Table 5.15. Each workload is repeated 30 times and its mean value are used to generate tables and graphs that are presented in Chapter 6.

Table 5.15: List of Identified Independent Workloads

| Benchmark Number | Benchmark Value |
|---|---|
| 1 | 9000 |
| 2 | 10800 |
| 3 | 12600 |
| 4 | 14400 |
| 5 | 16200 |
| 6 | 18000 |
| 7 | 19800 |
| 8 | 21600 |
| 9 | 23400 |
| 10 | 25200 |

Sieve benchmark considers execution time only (Jain, 2008) and one can compare performance of systems referring to their execution time. This principle in Sieve does not impact on our energy performance evaluation because energy in mobile devices is a direct function of execution time and data volume (Perrucci et al., 2011; Vallina-Rodriguez & Crowcroft, 2013). Moreover, to the best of our knowledge the only energy tool in Microsoft Windows is PowerCFG that does not profile energy consumption of the device, but only shows the intensive applications at every profiling window. Therefore, we limit our data collection to time only.

### *5.3.1 (a) Data Generation*

For data collection, we run the ASP and PHP codes separately on our testbed each one in two execution modes; local and PMC. In the first run, we execute our benchmark

using ASP on mobile device and run the entire task locally and collect time data. Then, we repeat the same experiments using our framework and collected the time results. The same set of experiments are repeated for PHP code. Once natively on the device and once using our own framework. The results of our experiments are presented in next chapter.

## 5.4 Comparative Study

In order to demonstrate efficiency and lightweight feature of our proposed framework, we have performed a comparative study on our application development and intensive-code execution mechanisms used in our framework with the offloading mechanism that is dominantly used in the literature, including (Chun et al., 2011; Satyanarayanan et al., 2009).

In typical mobile augmentation frameworks, the RMA is analyzed to identify the resource-intensive components. Once identified, the application is partitioned and intensive components are separated for offloading. At run-time, the code along with memory state and user input and preferences are offloaded to the remote server for execution. Once the results are back the framework reintegrates the results to the application, synchronizes the memory state with new results and continue local execution of application.

However, the computing overhead of analyzing the application, partitioning intensive code, and offloading the content (including code, memory state, and user input and preferences) is remarkably high that can surpass the performance gain (Sharifi et al., 2011). In Cloudcloud (Chun et al., 2011) the author create a virtual machine including the mobile operating system, memory state, and the entire application and offload it from mobile to the remote cloud. In cloud side, the virtual machine is installed, its execution is initiated and the application starts execution. However, the clone of mobile execution environment made by the CloneCloud framework is large and transferring it to the cloud is associ-

ated with long WAN latency that degrade application response time and energy efficiency (Sharifi et al., 2011). Several efforts after Clonecloud including Cloudlet (Satyanarayanan et al., 2009) aimed at shrinking the data volume of offloading. Though the achievements are remarkable, still the overhead of transmitting application code from mobile to the remote cloud-based resources is high (Sharifi et al., 2011; Shiraz et al., 2012). Authors in Cloudlet offered a novel VM overlay creation that is a partial VM of the mobiel device to minimize the amount of data to be transmitted. However, the minimum data volume for the VM overlay is reported more than 63 MB. Though the authors used a 100 MbPS wireless link between the mobile and cloudlet, creating, compressing, and transferring VM overlay to the cloudlet, and decompressing and applying it in the server side takes significant amount of time reported in the article itself.

However, our framework proposes a RESTful service-based mobile augmentation model in which applications are loosely coupled and do not require analysis and code partitioning. We also used ROA to enable mobile service providers to install the code prior the service delivery. In such an execution mechanism, at run-time the intensive parts of the application can be RESTfully called for execution in the remote server without need to transfer the code.

In this comparative study, we experimentally demonstrate that offloading the intensive component of RMA is a heavy weight and our framework is lightweight in the absence of code offloading.

### 5.4.1 Experiment Setup

In order to demonstrate the lightweight feature of our framework, we compare application execution mechanism deployed in our framework with application execution mechanism used in Cloudlet as a credible framework.

For our comparative study, we have build a real test-bed, including one mobile device as client and three mobile devices as service providers. Mobile devices used as MSP in this experiment feature 2.5 GHz clock speed processor, 4GB RAM, 1 TB storage running 64-bit Windows 7 Professional that serve as MSPs. The mobile client features Intel Core 2 Duo T7250 CPU, 4 GB RAM, and 160 GB storage running windows 7 Ultimate that plays the role of MSC. The wireless communication access point used in this experiment is a Cisco Linksys WRT54GL Broadband Router building a communication platform to connect MSC and MSPs via 2.4 GHZ band of 802.11 technology.

We choose to compare execution performance of our framework with Cloudlet using Sieve benchmark which is a highly popular and standard benchmark for execution performance evaluation of computing systems (Jain, 2008). For experiment, 10 workloads are selected after several rounds of pilot tests and their execution time are measured in two different execution modes. In the first execution mode, we use our framework to call remote execution of the Sieve algorithm execution. In second set of experiments, we followed the mechanism used in Cloudlet and transferred the code along with every call to the remote server. It is noteworthy that it is no practical to fully compare our framework with the work being done in Cloudlet since Cloudlet is a VM-based solution and our framework is VM-less. Thus, in the absence of VM, it is not practical to create overlay VM of the mobile run-time environment and forward it to the server. Instead, we apply the fundamental mechanism of offloading the user input along with the run-time environment to the remote server for every call.

Therefore, in second execution mode of this study, the run-time environment, including the compiler and benchmark values are transferred from the mobile to each remote server. The minimum VM overlay size of the run-time environment built in the Cloudlet is reported by the author as more than 63 MB for the smallest application. However, our

run-time environment is 8.9 MB. Thus, at any remote call, we transfer 8.9 MB data from mobile to each of the remote MSPs along with the code and user input (Sieve benchmark in this experiment). Once the results are ready in the server, we receive the results and reintegrated them with the client code. It is noteworthy that we do not require VM in the server side since our framework is built according to the service oriented architecture and RESTful services.

For each benchmark, we collect 30 data and present the results using 99 % confidence interval for the sake of reliability. We also performed paired sample T-test to demonstrate that the difference between results in two execution modes are statistically significant. Correlation analysis between the benchmark and corresponding results are also presented to demonstrate that how change in benchmark values change the execution time in each execution mode.

The results of out comparative study is presented in Section 6.4 in Chapter 6.

## 5.5 Statistical Data Analysis Method

In order to assess reliability and validity of our research, we perform several statistical analyses on primary data generated via benchmarking and statistical modeling. In the following section, we describe each of the statistical methods being used in this research.

### 5.5.1 Descriptive Statistics

In order to analyze data, describe improvements, and highlight significance of achievements in execution time and consumed energy for local and PMC modes, descriptive statistics including minimum, maximum, and mean are determined using IBM SPSS 22. Desired descriptive data are obtained for both data collected using benchmarking and mathematical modeling and are summarized in tabular and graphical presentations to fulfill the desired objectives.

### 5.5.2 Confidence interval

We present the execution time and consumed energy results of benchmarking experiments via interval estimate to enhance the reliability of our estimations. For reliability assurance, we iterate the data collection of each workload for 30 times. Instead of presenting point estimate for corresponding execution time and energy consumption of each workload, we use 99 % interval estimate. Therefore, we raise the confidence and reliability of results up to 99% when reporting the time and energy results.

### 5.5.3 Paired Samples T-Test

In this research, we use Paired Samples T-Test to ensure that there is a significant difference between the mean values of the identical measurement made in two different execution modes (local vs PMC). In our study, local execution time and PMC execution time values are paired data of the same workloads executed in two varied modes. Similarly, the local consumed energy and PMC consumed energy values are paired energy data of similar workloads in dissimilar modes. We use this test to ensure if execution modes (local and PMC) have significant impacts on time and energy or not. In other word, with the help of the results from Paired Samples T-Test we can show that execution time and consumed energy values in local and PMC modes have significant differences.

## 5.6 Conclusions

In this chapter, we describe the evaluation procedure in two parts of benchmarking and statistical modeling. In each section, the detailed description of data generation process for execution time and consumed energy are described. In statistical modeling section, we use observation-based analysis and employed independent replication method to devise the models for execution time and energy consumption in both local and PMC execution modes. The devised models are validated through split-sample approach and

the results of validation are reported. The results of performance evaluation are presented in next chapter that will be used to signify the strength and weaknesses our proposed framework.

# CHAPTER 6

# RESULTS AND DISCUSSION

In this chapter, we present results of our performance evaluation of the proposed model by analyzing two system-level metrics, namely execution time and energy consumption of the device for execution of the RMAs using series of benchmarking experiments. The evaluation results are validated via statistical modeling built using independent replication of new dataset.

The remainder of this chapter is as follows. Section 6.1 presents our benchmarking results and reports application execution time and energy consumption of the RMAs in local and Proximate Mobile Cloud (PMC) execution modes. The results of our statistical modeling are presented in section 6.2. Discussions and synthesis of the results are presented in section 6.6 and the chapter is concluded in section 6.7.

## 6.1 Performance Evaluation Results

Results of performance evaluation generated via benchmarking analysis are presented in this section in two parts. In the first part, data related to execution time analysis are presented followed by consumed energy analysis in part two. The benchmarking analysis is performed to evaluate the performance of the proposed framework.

### 6.1.1 Execution Time

This section presents temporal results of executing RMAs in two execution environments. One environment is local in which the entire computations, including intensive and non-intensive are executed on mobile device whereas in the other environment which is PMC, the application execution starts locally and execution of intensive tasks are per-

formed remotely using cloud of nearby mobile devices. Data related to execution time in this section are gathered using benchmarking analysis. Several Tables and charts are used to demonstrate the findings.

Tables 6.1 and 6.2 present the temporal data related to application execution time collected in local and PMC execution modes, respectively. Each of the tables presents mean execution time, standard deviation, error estimate, and execution time of 30 workloads in three intensity levels with 99 % confidence interval. Column *N* in the Table represents the number of workloads whose execution times are used to calculate mean values of the minimum, maximum, and mean represented in the Table's columns. For instance, $N = 10$ shows that values shown in minimum, maximum, and mean columns are mean values of execution time for 10 workloads.

The small *error estimates* shown in the *fifth* column in the Table 6.1 and 6.2 ensure reliability of the collected data during real time experimentation. The minimum, maximum, and mean error estimate values for execution times are 0.00145, 0.01133, 0.0042 for local execution and 0.01135, 0.040107, 0.025 for PMC execution, respectively. For example, the maximum error estimate for the $6^{th}$ workload executed in PMC execution time with 99 % confidence interval is $3056.2(+/-)145$ meaning that the PMC execution time of the workload falls in the range of $(3056.2-145) < \mu < (3056.2+145)$ or $2911.2 < \mu < 3201.2$. This range shows that if the execution is repeated, the execution time value falls in this range with 99% confidence. To better demonstrate the significance of our achievements and effectively interpret the results, we perform comprehensive statistical analysis which is presented as follows.

Descriptive statistics of results in local and PMC execution modes, including minimum, maximum, and mean execution time of workloads are summarized in Table 6.3 in three intensity levels of low, medium, and high beside the mean execution time of all

Table 6.1: Execution Time with 99% Confidence Interval in Local Execution Mode Generated via Benchmarking

| Workload#* | Intensity | Execution Time (ms) | Standard Deviation | Error Estimate | 99% Confidence Interval Execution Time |
|---|---|---|---|---|---|
| 1 | Low | 6102.2 | 142.2 | 67 | 6102.2(+/-)67 |
| 2 | | 6960.9 | 117.3 | 55.3 | 6960.9(+/-)55.3 |
| 3 | | 7759 | 144.1 | 67.9 | 7759(+/-)67.9 |
| 4 | | 8340.4 | 200.7 | 94.5 | 8340.4(+/-)94.5 |
| 5 | | 9160.6 | 197.5 | 93 | 9160.6(+/-)93 |
| 6 | | 9898.9 | 162.5 | 76.6 | 9898.9(+/-)76.6 |
| 7 | | 10959.4 | 165.3 | 77.9 | 10959.4(+/-)77.9 |
| 8 | | 12146.6 | 172.2 | 81.1 | 12146.6(+/-)81.1 |
| 9 | | 12972.2 | 141.4 | 66.6 | 12972.2(+/-)66.6 |
| 10 | | 13869.5 | 162.8 | 76.7 | 13869.5(+/-)76.7 |
| 11 | Medium | 25601.7 | 138.5 | 65.3 | 25601.7(+/-)65.3 |
| 12 | | 26927 | 107.3 | 50.5 | 26927(+/-)50.5 |
| 13 | | 28329.1 | 124.8 | 58.8 | 28329.1(+/-)58.8 |
| 14 | | 29734.6 | 203.8 | 96 | 29734.6(+/-)96 |
| 15 | | 30939.7 | 192 | 90.4 | 30939.7(+/-)90.4 |
| 16 | | 32658.3 | 183.8 | 86.6 | 32658.3(+/-)86.6 |
| 17 | | 34340.4 | 207 | 97.5 | 34340.4(+/-)97.5 |
| 18 | | 35636.7 | 220 | 103.6 | 35636.7(+/-)103.6 |
| 19 | | 37514.1 | 168.2 | 79.2 | 37514.1(+/-)79.2 |
| 20 | | 40067.9 | 247 | 116.3 | 40067.9(+/-)116.3 |
| 21 | High | 49015.3 | 230.7 | 108.7 | 49015.3(+/-)108.7 |
| 22 | | 55924 | 263.3 | 124 | 55924(+/-)124 |
| 23 | | 63954.8 | 277 | 130.5 | 63954.8(+/-)130.5 |
| 24 | | 72544.3 | 239.9 | 113 | 72544.3(+/-)113 |
| 25 | | 80900.2 | 304.9 | 143.6 | 80900.2(+/-)143.6 |
| 26 | | 88915.7 | 293.8 | 138.4 | 88915.7(+/-)138.4 |
| 27 | | 99765.5 | 374.4 | 176.3 | 99765.5(+/-)176.3 |
| 28 | | 106987.1 | 313.5 | 147.7 | 106987.1(+/-)147.7 |
| 29 | | 116875.2 | 371.7 | 175.1 | 116875.2(+/-)175.1 |
| 30 | | 127080.4 | 372.5 | 175.5 | 127080.4(+/-)175.5 |

Workload values are presented in Table 5.1

Table 6.2: Execution Time with 99% Confidence Interval in PMC Execution Mode Generated via Benchmarking

| Workload#* | Intensity | Execution Time (ms) | Standard Deviation | Error Estimate | 99% Confidence Interval Execution Time |
|---|---|---|---|---|---|
| 1 | Low | 2298.6 | 175.75 | 82.8 | 2298.6(+/-)82.8 |
| 2 | | 2536.06 | 211.84 | 99.8 | 2536.1(+/-)99.8 |
| 3 | | 2591.93 | 200.07 | 94.2 | 2591.9(+/-)94.2 |
| 4 | | 2673.4 | 267.76 | 126.1 | 2673.4(+/-)126.1 |
| 5 | | 2881.1 | 271.03 | 127.7 | 2881.1(+/-)127.7 |
| 6 | | 3056.16 | 307.79 | 145 | 3056.2(+/-)145 |
| 7 | | 3189.97 | 276.28 | 130.1 | 3190(+/-)130.1 |
| 8 | | 3314.83 | 284.411 | 134 | 3314.8(+/-)134 |
| 9 | | 3465.46 | 247.86 | 116.8 | 3465.5(+/-)116.8 |
| 10 | | 3592.46 | 199.07 | 93.8 | 3592.5(+/-)93.8 |
| 11 | Medium | 4329.93 | 227.79 | 107.3 | 4329.9(+/-)107.3 |
| 12 | | 4621.667 | 340.08 | 160.2 | 4621.7(+/-)160.2 |
| 13 | | 4698.6 | 277.28 | 130.6 | 4698.6(+/-)130.6 |
| 14 | | 4809.9 | 190.99 | 90 | 4809.9(+/-)90 |
| 15 | | 4891.16 | 150.26 | 70.8 | 4891.2(+/-)70.8 |
| 16 | | 5188.73 | 243.15 | 114.5 | 5188.7(+/-)114.5 |
| 17 | | 5387.2 | 248.50 | 117.1 | 5387.2(+/-)117.1 |
| 18 | | 5675.6 | 251.99 | 118.7 | 5675.6(+/-)118.7 |
| 19 | | 5770.2 | 226.58 | 106.7 | 5770.2(+/-)106.7 |
| 20 | | 2960 | 231.62 | 109.1 | 5793.1(+/-)109.1 |
| 21 | High | 5979.47 | 253.24 | 119.3 | 5979.5(+/-)119.3 |
| 22 | | 6134.57 | 207.75 | 97.9 | 6134.6(+/-)97.9 |
| 23 | | 7414.03 | 280.87 | 132.3 | 7414(+/-)132.3 |
| 24 | | 7942.13 | 265.56 | 125.1 | 7942.1(+/-)125.1 |
| 25 | | 8551.57 | 304.98 | 143.7 | 8551.6(+/-)143.7 |
| 26 | | 8657.93 | 312.72 | 147.3 | 8657.9(+/-)147.3 |
| 27 | | 9156.6 | 322.78 | 152 | 9156.6(+/-)152 |
| 28 | | 9862.97 | 274.12 | 129.1 | 9863(+/-)129.1 |
| 29 | | 10395.63 | 300.67 | 141.6 | 10395.6(+/-)141.6 |
| 30 | | 10939.17 | 263.56 | 124.2 | 10939.2(+/-)124.2 |

Workload values are presented in Table 5.1

intensity levels. As descriptive statistics in the Table shows, executing the task on nearby mobile devices instead of local device can reduce the execution time as significant as 91.39%.

In low, medium, and high intensity levels, the minimum time savings are as high as 62.33%, 85.9%, 87.8% which are increasing as the workload intensities raise. The maximum time saving for low, medium, and high intensity workloads are approximately 68.7%, 85%, and 91.4% respectively. The mean time saving of low, medium, and high intensity workloads are about 69.8%, 84%, and 90% respectively. For instance, in low intensity workloads, the local execution of the $5^{th}$ workload takes 9160.6$ms$. However, executing the same workload in PMC resources needs only 2881.1$ms$. The amount of time saved is as high as 68.55% i.e., 6279.5$ms$. In medium intensity and high intensity where computation volume is bigger, the time saved for execution of $15^{th}$ and $25^{th}$ workloads are 26048.13$ms$ and 72348.63$ms$ which are as high as 84.19% and 89.43 respectively. As the numerical values suggest, the time saving in higher intensity levels are more than lower intensity workloads.

The mean execution time of workloads in local execution mode, for the low intensity category, is as much as 3.3 times more than PMC execution which is remarkably high despite of low intensity of the workloads. Similarly, for medium and high intensity levels, the mean execution time of workloads in the mobile devices is as high as 6.3 and 10.13 times more than PMC execution mode, respectively.

As described in previous chapter, execution of each workload is repeated 30 times to enhance reliability of performance evaluation. So, data plotted in Figure 6.1 are mean execution time of the workloads for both local and PMC execution modes. Each blue diagonal stripped bar in the Figure, represents the mean value of local execution time of 30 iterations for each corresponding workload. Similarly each green patterned bar represents

Table 6.3: Descriptive Statistics of Execution Time Data Generated via Benchmarking

| **Description** | **Intensity** | **N** | **Minimum(ms)** | **Maximum (ms)** | **Mean(ms)** |
|---|---|---|---|---|---|
| Local Execution Time | Low | 10 | 6102.20 | 13869.50 | 9816.97 |
| PMC Execution Time | | 10 | 2298.60 | 4329.93 | 2960 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Time | Medium | 10 | 25601.70 | 40067.90 | 32174.95 |
| PMC Execution Time | | 10 | 3592.47 | 5979.47 | 5116.61 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Time | High | 10 | 49015.30 | 127080.40 | 86196.2500 |
| PMC Execution Time | | 10 | 5979.47 | 10939.17 | 8503.4070 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Time | All | 30 | 6102.20 | 127080.40 | 42729.39 |
| PMC Execution Time | | 30 | 2298.60 | 10939.17 | 5526.67 |
| Valid N (listwise) | | 30 | | | |

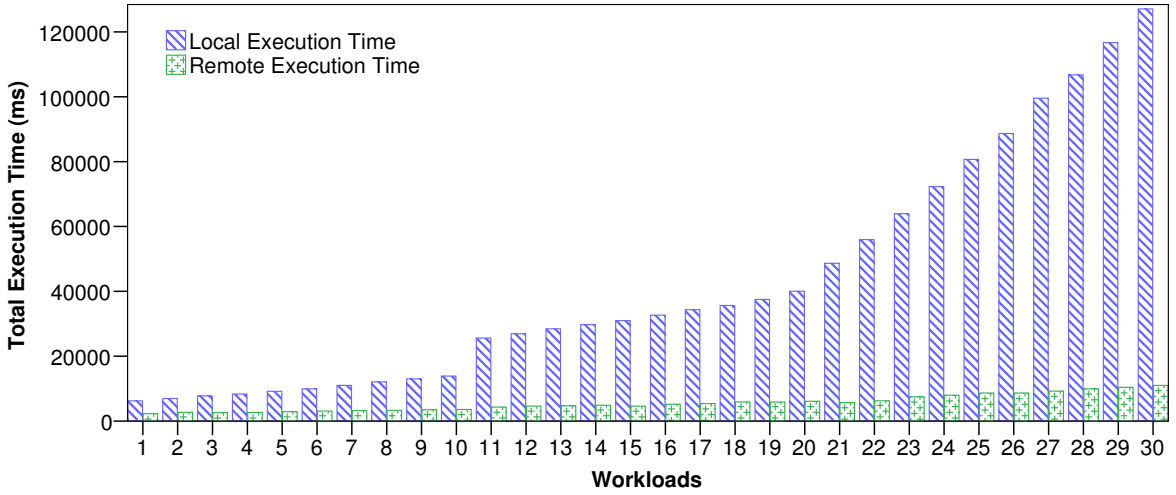



Figure 6.1: Execution Time for 30 Workloads Generated via Benchmarking: Local vs PMC Execution

the mean value of PMC execution time of 30 iterations for each corresponding workload. The graph clearly depicts increasing complexity in low, medium, and high intensity levels from left to right. Growth of the workloads has significant impact on the execution times when the workloads are entirely executed on the mobile device. However, the growth rate in PMC execution mode is remarkably smaller than local execution. Execution of the last workload in our experiment takes more than 127000*ms* to complete, which suggest incapacitation of executing higher workloads in the mobile device. Although the results of descriptive statistics summarized in Tables 6.1, 6.2, 6.3 and demonstrated in Figure 6.1

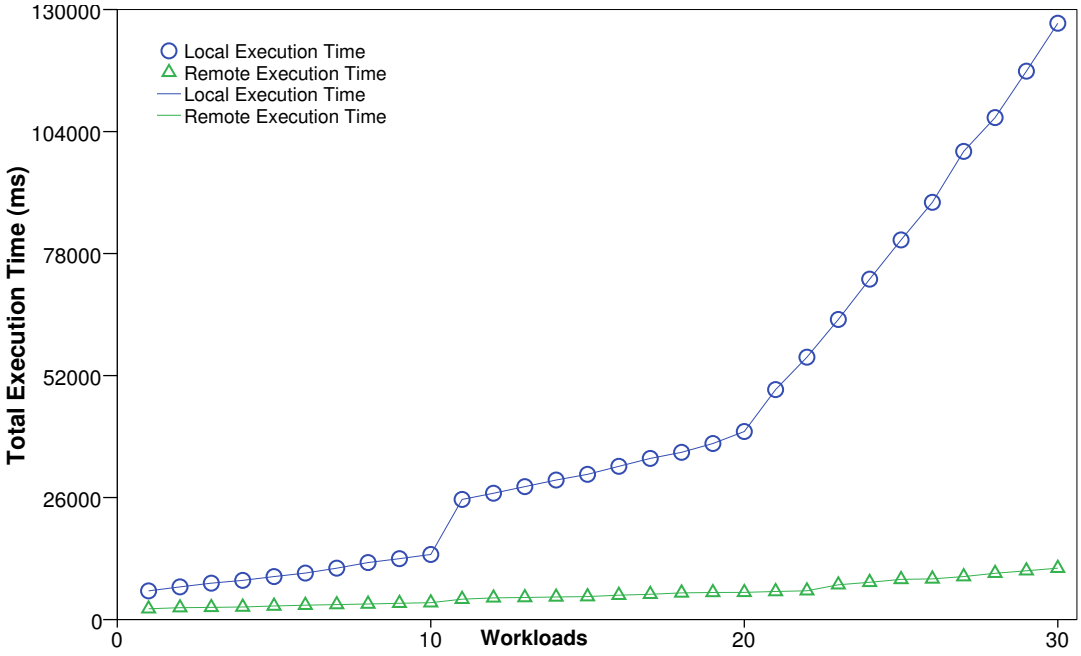



Figure 6.2: Scattered Plot with Interpolation Lines for Application Execution Time

and 6.2 advocate remarkable improvement in execution time of the application in PMC, further analysis is undertaken via paired samples T-Test to ensure that the mean application execution times in local and PMC modes are significantly different. The results of paired samples T-Test are presented in Table 6.4.

As the Table 6.4 summarizes, we found that $t(29) = 6.104, p < 0.01$. The t-value and p-value (i.e., sigma) advocate significant differences between mean local and PMC execution time values. Positive t-value advocates that local execution takes more time than PMC on average. Therefore, the time saving using our proposed framework is significant compared to local execution mode.

For local execution of low intensity workloads, the native computing resources of the mobile device suffice to complete the task without much maintenance operations (e.g.,

Table 6.4: Results of Paired Samples T-Test for Analyzing the Significance of Execution Time Conservation in PMC Mode Compared to Local Mode.

| **Paired Sample Test** | | | **Paired Samples Correlation** |
|---|---|---|---|
| T | df | Sig | Sig (2-tailed) |
| 6.104 | 29 | 0.000 | 0.000 |

loading data into memory and interrupting CPU execution). In the beginning of the execution, all the data are loaded into the main memory and execution starts by the CPU. Once the execution completed, the results are sent back to the main memory to present to the user. Such operation does not entertain unnecessarily I/O tasks. However, when workload intensity is high, computation in local execution mode demands more resources, including CPU, cache, RAM, and storage which are not available natively. Such constraints cause execution to prolong. For instance, in the absence of high clock speed CPU, the execution takes more CPU cycle to complete. Moreover, due to limitation in the main memory, it is not possible to store the entire data into the RAM for medium and high intensity workloads. Thus, there will be continuous time-consuming I/O operations to load data into the main memory and store them into the peripheral RAM (storage) and vice versa. Such switching and I/O operations are highly contributing to the execution time prolonging in higher workloads. In order to mitigate the anomaly caused by low RAM in execution of the application, we restart the device for every few executions when collecting real data for medium and high intensity workloads.

Nevertheless, significant differences in local and PMC execution enable mobile users to initiate PMC execution of extremely huge workloads on their mobile devices toward gaining similar functionality experience as desktop computers.

Such differences are better visible in Figure 6.2. Scattered circles and triangles across the graph and corresponding interpolating lines show the differences in achievements and the correlation between the workloads intensity and time saving. In the first ten workloads with low intensity, the difference between circle and triangle is comparatively smaller than of medium and high intensity.

Execution time in local mode highly is affected by workload intensity and computing power of the mobile device (including CPU clock speed, RAM, storage, cache). However,

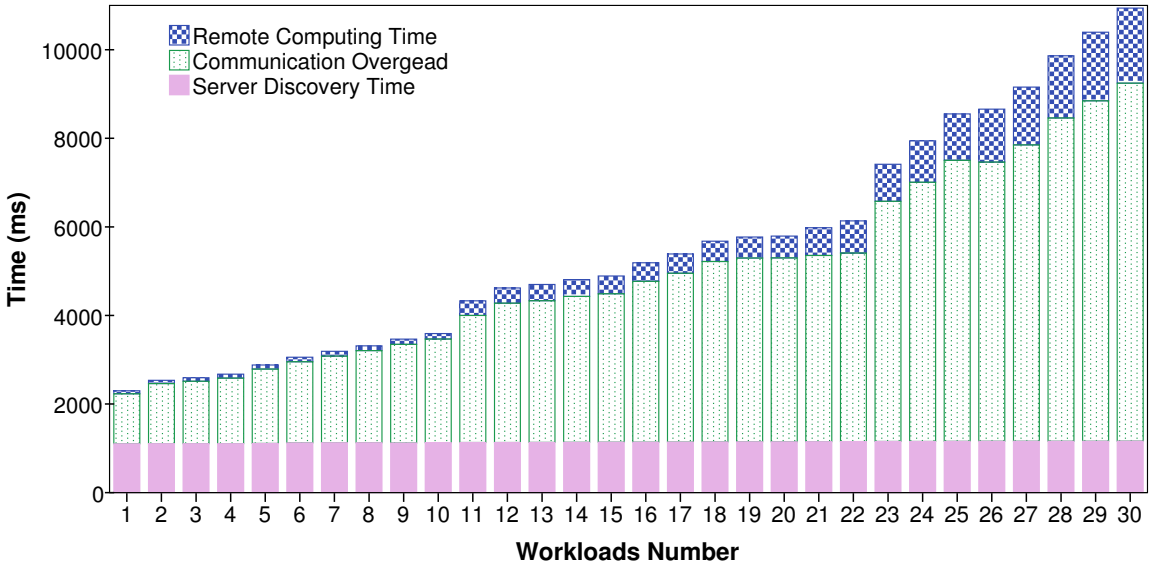



Figure 6.3: Breakdown of Remote Execution Time for 30 Workloads

the PMC execution time is dependent on other metrics, including computing and communication delay of discovering MSP (the time takes that MSC sends request and fetches the IP address(es) of MSP(s) from the TSG), wireless communication performance, and computing capability of the MSP(s). Figure 6.3 shows the detailed timing of each contributing entity to the total execution time in PMC execution mode. Pinkish solid chunks located in the bottom of the bars represent discovery delay of identifying potential MSPs in proximity. This delay includes the communication delay for the TSC to communication with TSG to find nearby MSP and also searching time into the TSG database of MSPs. The green dotted chunks located in middle of the bars whose values are the highest among others, show the communication latency when performing PMC execution calls. The checkered chunks are computing time of remote MSPs. As the results in the figure illustrate, the communication overhead and discovery delay are the highest values. Nevertheless, these delays does not have significant impact on the user experience due to asynchronous nature of communication between MSC and MSP that unobtrusively engages user with the application and enable distraction-free interaction with the device till the PMC response arrives. During MSP discovery time, MSC sends asynchronous request to the TSG and ask for po-

tential MSP(s) in proximity. On successful MSP discovery, the TSG sends the URI (IP addresses + resource ID) of desired resource in the identified MSP(s) to the corresponding MSC and execution continues.

In overall, the results of benchmarking on execution time show significant improvement in application responsiveness when our proposed model is deployed. This remarkable achievements are due to several factors including, lightweight nature of underlying ROA technology, low-overhead deployed client-server communication technology, short WAN latency due to low-hop MSC-MSP communications, and homogeneity of the MSC and MSP which are significant characteristics that are considered in design and development of the proposed framework. The results in this section are comparable with and supporting the findings in statistical analysis section.

### 6.1.2 Consumed Energy

In this section, we present energy results of executing application on local and PMC execution modes along with statistical comparison. Tables 6.5 and 6.6 present the data related to the energy consumed by the mobile device which are collected in local and PMC execution modes for three intensity levels of low, medium and high, respectively. Each table summarizes mean consumed energy, standard deviation, error estimate, and consumed energy with 99 % confidence interval for 30 different workloads in three intensity levels. Column *N* in the Table represents the number of workloads whose energy consumption are used to calculate mean values of the minimum, maximum, and mean represented in the Table's columns.

Similar to the execution time, we present consumed energy with 99% confidence interval to enhance reliability of our data. The small value of error estimates in the *fifth* columns in both Tables 6.5 and 6.6 testify reliability of collected energy data. The min-

Table 6.5: Consumed Energy values with 99% Confidence Interval For 30 Workloads in Local Execution Mode

| Workload No.* | Intensity | Consumed Energy (mJ) | Standard Deviation | Error Estimate | 99% Confidence Interval Consumed Energy |
|---|---|---|---|---|---|
| 1 | Low | 1747.9 | 46.81467 | 22.1 | 1747.9(+/-)22.1 |
| 2 | | 2043.2 | 73.81029 | 34.8 | 2043.2(+/-)34.8 |
| 3 | | 2240.6 | 37.77643 | 17.8 | 2240.6(+/-)17.8 |
| 4 | | 2468.9 | 31.56537 | 14.9 | 2468.9(+/-)14.9 |
| 5 | | 2680.5 | 83.54829 | 39.4 | 2680.5(+/-)39.4 |
| 6 | | 2914.7 | 163.5267 | 77 | 2914.7(+/-)77 |
| 7 | | 3235.9 | 72.49606 | 34.1 | 3235.9(+/-)34.1 |
| 8 | | 3574.3 | 32.36537 | 15.2 | 3574.3(+/-)15.2 |
| 9 | | 3850 | 42.20149 | 19.9 | 3850(+/-)19.9 |
| 10 | | 4073 | 135.1696 | 63.7 | 4073(+/-)63.7 |
| 11 | Medium | 7355 | 45.79143 | 21.6 | 7355(+/-)21.6 |
| 12 | | 7686.1 | 110.847 | 52.2 | 7686.1(+/-)52.2 |
| 13 | | 8017.1 | 72.09484 | 34 | 8017.1(+/-)34 |
| 14 | | 8423.6 | 128.7653 | 60.7 | 8423.6(+/-)60.7 |
| 15 | | 8820.1 | 168.3503 | 79.3 | 8820.1(+/-)79.3 |
| 16 | | 9384.5 | 96.78691 | 45.6 | 9384.5(+/-)45.6 |
| 17 | | 9763.5 | 59.06478 | 27.8 | 9763.5(+/-)27.8 |
| 18 | | 10086.9 | 93.54711 | 44.1 | 10086.9(+/-)44.1 |
| 19 | | 10693.5 | 60.45868 | 28.5 | 10693.5(+/-)28.5 |
| 20 | | 11202 | 49.76455 | 23.4 | 11202(+/-)23.4 |
| 21 | High | 13413.3 | 63.86038 | 30.1 | 13413.3(+/-)30.1 |
| 22 | | 15466.3 | 37.30064 | 17.6 | 15466.3(+/-)17.6 |
| 23 | | 17789.1 | 185.4738 | 87.4 | 17789.1(+/-)87.4 |
| 24 | | 19742.2 | 114.0397 | 53.7 | 19742.2(+/-)53.7 |
| 25 | | 21840.4 | 127.8388 | 60.2 | 21840.4(+/-)60.2 |
| 26 | | 24386.6 | 123.1188 | 58 | 24386.6(+/-)58 |
| 27 | | 27209.3 | 173.7555 | 81.8 | 27209.3(+/-)81.8 |
| 28 | | 29217.6 | 119.1202 | 56.1 | 29217.6(+/-)56.1 |
| 29 | | 31971.5 | 123.8137 | 58.3 | 31971.5(+/-)58.3 |
| 30 | | 34838.2 | 185.5017 | 87.4 | 34838.2(+/-)87.4 |

Workload values are presented in Table 5.1

Table 6.6: Consumed Energy values with 99% Confidence Interval For 30 Workloads in PMC Execution Mode

| **Workload No.*** | **Intensity** | **Consumed Energy (mJ)** | **Standard Deviation** | **Error Estimate** | **99% Confidence Interval Consumed Energy** |
|---|---|---|---|---|---|
| 1 | Low | 1301.7 | 92.4 | 43.5 | 1301.7(+/-)43.5 |
| 2 | | 1466.3 | 67.8 | 31.9 | 1466.3(+/-)31.9 |
| 3 | | 1548.8 | 129.2 | 60.9 | 1548.8(+/-)60.9 |
| 4 | | 1597.9 | 123.3 | 58.1 | 1597.9(+/-)58.1 |
| 5 | | 1709.1 | 104.6 | 49.3 | 1709.1(+/-)49.3 |
| 6 | | 1822.4 | 137.9 | 64.9 | 1822.4(+/-)64.9 |
| 7 | | 1902.0 | 62.5 | 29.4 | 1902(+/-)29.4 |
| 8 | | 2017.7 | 106.1 | 50.0 | 2017.7(+/-)50 |
| 9 | | 2032.4 | 121.8 | 57.4 | 2032.4(+/-)57.4 |
| 10 | | 2153.9 | 135.6 | 63.9 | 2153.9(+/-)63.9 |
| 11 | Medium | 2618.8 | 119.0 | 56.0 | 2618.8(+/-)56 |
| 12 | | 2751.1 | 115.6 | 54.4 | 2751.1(+/-)54.4 |
| 13 | | 2904.7 | 146.0 | 68.8 | 2904.7(+/-)68.8 |
| 14 | | 2976.1 | 127.2 | 59.9 | 2976.1(+/-)59.9 |
| 15 | | 3178.1 | 142.2 | 67.0 | 3178.1(+/-)67 |
| 16 | | 3160.7 | 90.2 | 42.5 | 3160.7(+/-)42.5 |
| 17 | | 3279.8 | 106.2 | 50.0 | 3279.8(+/-)50 |
| 18 | | 3453.7 | 134.5 | 63.4 | 3453.7(+/-)63.4 |
| 19 | | 3544.2 | 131.8 | 62.1 | 3544.2(+/-)62.1 |
| 20 | | 3566.5 | 145.3 | 68.5 | 3566.5(+/-)68.5 |
| 21 | High | 3813.1 | 124.6 | 58.7 | 3813.1(+/-)58.7 |
| 22 | | 3966.5 | 103.6 | 48.8 | 3966.5(+/-)48.8 |
| 23 | | 4473.0 | 188.3 | 88.7 | 4473(+/-)88.7 |
| 24 | | 4865.2 | 140.8 | 66.3 | 4865.2(+/-)66.3 |
| 25 | | 5328.1 | 137.1 | 64.6 | 5328.1(+/-)64.6 |
| 26 | | 5395.0 | 165.9 | 78.2 | 5395(+/-)78.2 |
| 27 | | 5696.0 | 165.0 | 77.7 | 5696(+/-)77.7 |
| 28 | | 6190.1 | 178.3 | 84.0 | 6190.1(+/-)84 |
| 29 | | 6361.2 | 136.1 | 64.1 | 6361.2(+/-)64.1 |
| 30 | | 6585.5 | 173.0 | 81.5 | 6585.5(+/-)81.5 |

Workload values are presented in Table 5.1

Table 6.7: Descriptive Statistics of Consumed Energy Data Generated via Benchmarking

| Description | Intensity | N | Minimum | Maximum | Mean |
|---|---|---|---|---|---|
| Local Execution Energy | Low | 10 | 1747.90 | 4073 | 2882.81 |
| PMC Execution Energy | | 10 | 1301.70 | 2153.90 | 1755.22 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Energy | Medium | 10 | 7355 | 11202.00 | 9143.23 |
| PMC Execution Energy | | 10 | 2618.80 | 3566.50 | 3143.37 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Energy | High | 10 | 13413.30 | 34838.20 | 23587.45 |
| PMC Execution Energy | | 10 | 3813.10 | 6585.50 | 5267.37 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Energy | All | 30 | 1747.90 | 34838.20 | 11871.16 |
| PMC Execution Energy | | 30 | 1301.70 | 6585.50 | 3388.65 |
| Valid N (listwise) | | 30 | | | |

imum, maximum, and mean estimate errors are 0.00113, 0.02641, and 0.00646 respectively.

Descriptive statistics of analyzing consumed energy data are summarized in Table 6.7 including minimum, maximum, and mean consumed energy in three intensity levels of low, medium, and high in addition to the mean energy consumed for all intensity levels. As shown in the Table, there is significant energy saving when performing task outside the mobile on the cloud of nearby mobile devices. The achievements has direct correlation with workload intensity, meaning that growth in workload's intensities increases the achievements. Such tendency suggests to not offload very low intensity tasks. PMC execution reduces mobile consumed energy as significant as 81% compared to the local execution. In average, consumed energy is saved approximately 71.45% that testifies the fact that local execution of the RMA tasks consumes 3.5 times more energy compared to the PMC execution.

In low, medium, and high intensity levels, the least energy savings are 25.5%, 64.39%, 71.57% which are increasing as the workload intensities raise. The most energy saving for low, medium, and high intensity workloads are approximately 47%, 68%, and 81%

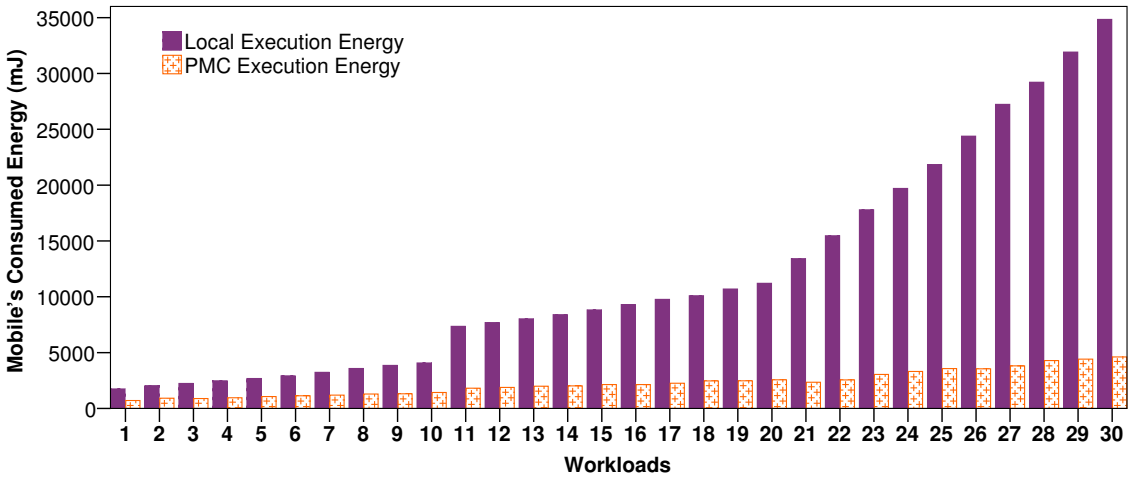



Figure 6.4: Mobile Consumed Energy for 30 Workloads: Local vs PMC Execution

respectively. In low intensity workloads, average energy consumption of executing the workloads in local and PMC mode are 2882.81 and 1755.22, respectively. Thus, the energy consumption is reduced as high as 39% $(i.e., 1124.3)$. The energy saving is remarkably higher in medium and high intensity levels; the difference in energy consumption of workloads in local and PMC modes are as high as 65.62%, and 77.67%, respectively that is significant achievement.

The mean energy saving of workloads in local execution mode, for the low intensity category is 1.64 times more than PMC execution. Similarly, for medium and high intensity levels, the mean energy consumed for the workloads in the mobile devices is as high as 2.9 and 4.47 times more than PMC execution mode, respectively.

Figures 6.4 and 6.5 illustrate comparative results of consumed energy in local and PMC execution modes, respectively. The solid bars in Figure 6.4 represent consumed energy of local execution mode and the patterned bars represent consumed energy of executing workloads in PMC mode. The significance of PMC execution is higher when the workload intensity is high. Although in low intensities there is noticeable energy saving, the PMC execution is more beneficial when the workloads are intense.

Further analysis of local and PMC consumed energy is performed via paired sam-

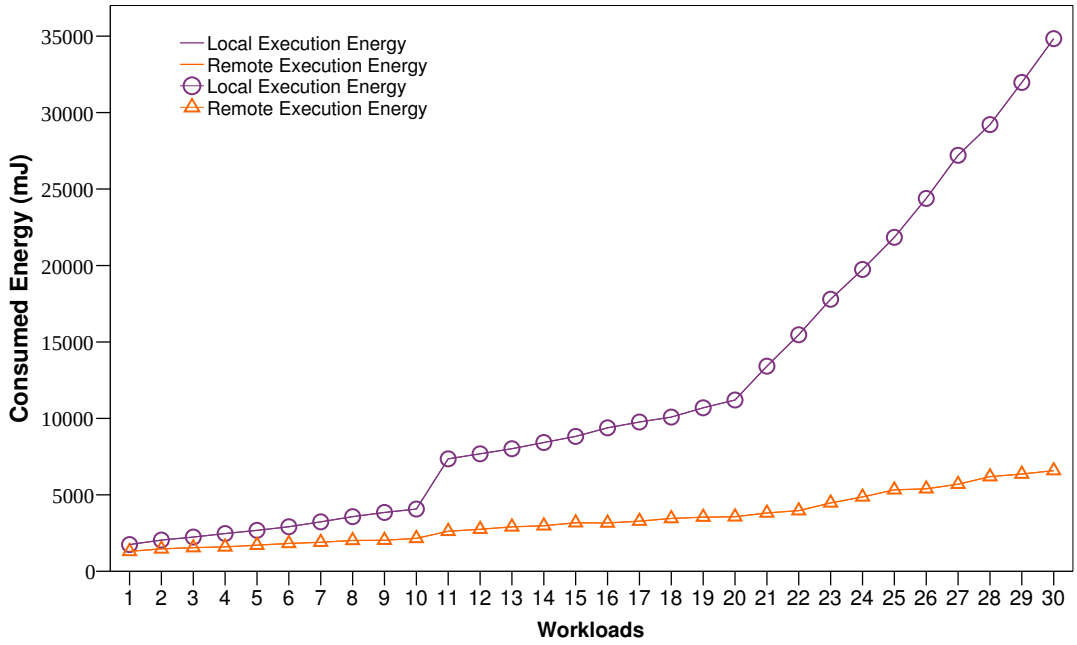



Figure 6.5: Scattered Plot with Interpolation Lines for Mobile Consumed Energy: Local vs PMC Mode

ples T-Test to demonstrate significance of energy saving of our proposal in benchmarking analysis. The results of paired samples T-Test are presented in Table 6.8.

Table 6.8: Results of Paired Samples T-Test for Analyzing the Significance of Energy Conservation in PMC Mode Compared to Local Mode.

| **Paired Sample Test** | | | **Paired Samples Correlation** |
|---|---|---|---|
| T | df | Sig | Sig (2-tailed) |
| 5.697 | 29 | 0.000 | 0.000 |

As the Table 6.8 represents, t(29)=5.697, p<0.01 which advocate significant differences between mean local and PMC energy consumptions. Positive t-value testifies that mean local energy consumption is larger than mean PMC energy usage. Therefore, the average energy saving using our proposed framework is significant compared to local execution mode.

It is also important to note that increase in energy consumption of the mobile device has several other slightly contributing factors beside the time. During the measurement of the consumed energy, we observed situations that energy consumption has significantly increased or decreases without much change in time. Upon thorough investigation, we

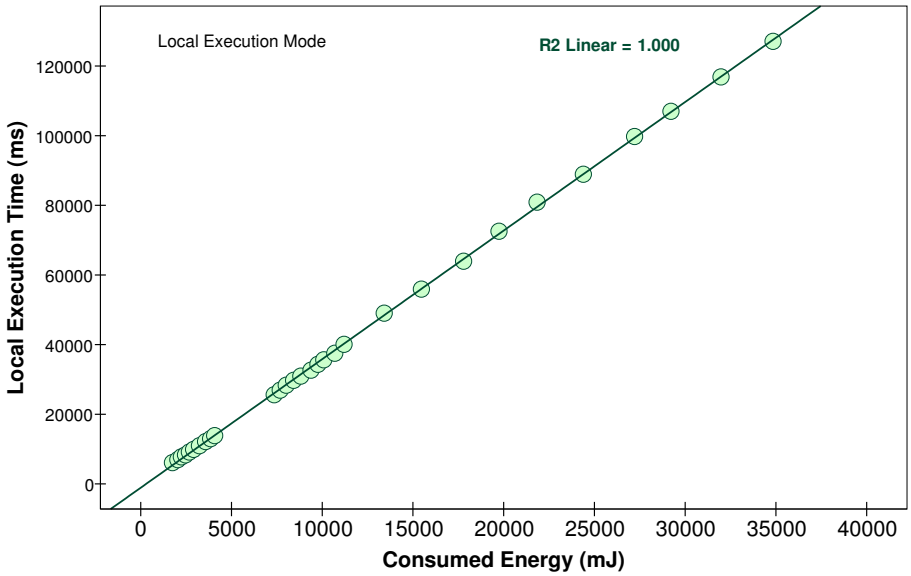



Figure 6.6: Linear Correlation Between Execution Time and Consumed Energy in Local Execution Mode

realized that change in energy consumption of the mobile device is a factor of several external factors, including the battery temperature, its current charge, and the device temperature. When battery is low, there is much higher energy consumption rather than when the battery has more than 50% charge. Therefore, during the entire data collection task, the battery remained at above 50% charge. Moreover, heat is another cause of excess energy consumption that was observed. When the CPU and battery are hot, there will be more energy consumption than when it is cool. Thus, to ensure integrity in data collection, all data are collected in alternative time to ensure the device temperature is low.

The results in Figure 6.5 highlight the correlation between workload intensity (increasing from left to right) and energy consumption of mobile device. In order to investigate this possible correlation, we perform further analysis to identify the existing correlation in both local and PMC execution. Since, the workload intensity impacts on the execution time with 100% correlation, the time is the most effective factor. So, we analyze time against energy and plot the results of this investigation in Figures 6.6 and 6.7. The $R^2$ values in the charts testify significant correlation between time and energy.

In Figure 6.6, there is very meaningful and significant linear correlation between exe-

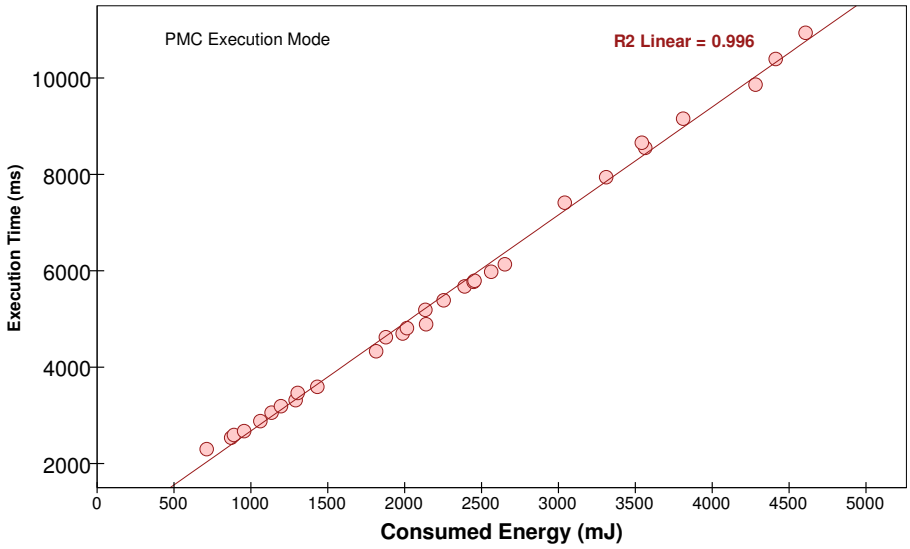



Figure 6.7: Linear Correlation Between Execution Time and Consumed Energy in Remote Execution

cution time and consumed energy which is due to the harmony in execution environment. In the local execution mode, the most influential factor that impact on execution time is the computational intensity which is measured by the amount of time that the processor is performing. Hence, the more computing time, the higher energy consumption. However, in PMC execution, there are other factors that very slightly reduce this linear correlation from 100% to 99.6%. In Figure 6.7, circles adjacency with the fit line is less tight compared to the Figure 6.6 which shows secondary factors, particularly communication overhead. Since, mobile devices utilize wireless communication to perform intercommunication, wireless network inconsistency and unpredictability (especially when data transmission volume is high) impact on the energy consumption of the mobile device. Figure 6.8 presents the percent of energy consumed by PMC computing and energy that is spent on wireless communication by Wi-Fi.

It is noteworthy that energy consumed by Wi-Fi is less (compared to the time consumed during wireless communication) because it contains only the energy that Wi-Fi radio transmitter consumed to send and receive the requests and responses. Thus, the energy consumed while computing is taking place outside the mobile device is not consid-

ered as Wi-Fi energy but is considered as PMC computing energy. The energy consumed for PMC executions shown by green diagonal stripped bars contain (1)energy consumed for the time taken as transmission delay to send request from the MSC to the MSP and receive the reply, (2) the energy consumed while the TSG is looking at its database to find IP addresses of nearby MSPs, (3) energy consumed for the time taken as transmission delay for MSC to perform calls to MSPs and the time taken for the results to return and (4) the energy consumed for the time taken to execute the longest task in MSP.

However, despite large number of factors affecting the application energy consumption in read world execution environments, in our proposed framework the distance between circles and the fit line are negligible due to small number of intermediate hops between mobile service consumer and mobile service provider that remarkably hike the framework significance and usefulness.

## 6.2 Validation Results

In order to validate the results of performance evaluation produced via benchmarking in section 6.1, statistical modeling is undertaken in this study whose results are presented in this section in two parts. Firstly, we provide data related to analysis of execution time

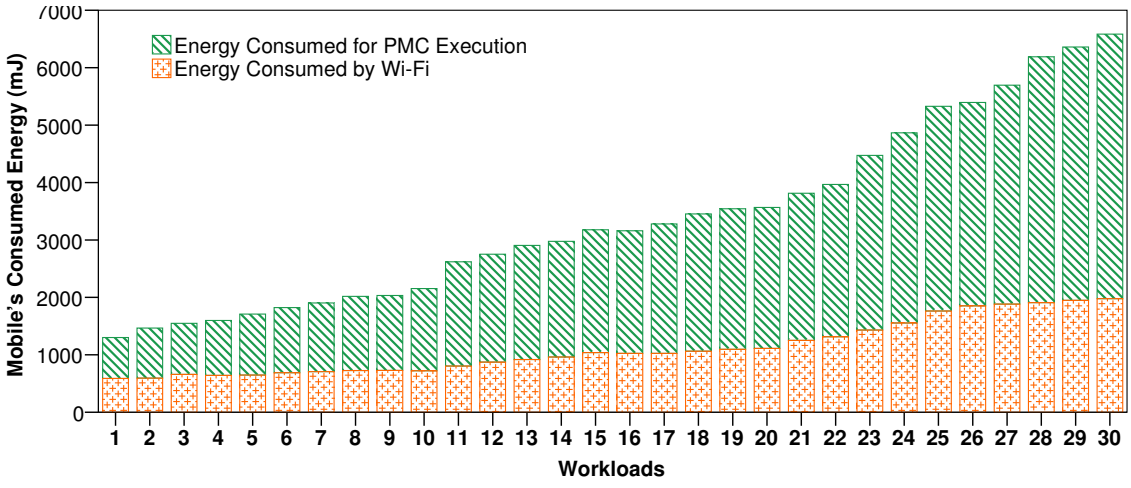



Figure 6.8: Proportional Energy Consumption Relation of PMC Computing and Wi-Fi in PMC Execution Mode

of 30 workloads in two execution modes of local and PMC. In second part, we report analytical data related to the consumed energy of 30 workloads in local and PMC execution modes. Descriptive statistics of our findings via statistical modeling are reported and their individual values are plotted using several figures. We use paired samples t-test results to confirm significance of our achievements. The results of the t-test ensure that there are significant differences between mean values of execution time and energy of local and PMC execution modes.

### 6.2.1 Execution Time

We have presented the execution time data in Table 6.9 for 30 workloads in three intensity levels of low, medium, and high collected in local and PMC execution modes. The results advocate significant achievement when performing resource-intensive task in PMC execution mode. We further provide -in Table 6.10- the descriptive statistics of data related to total execution time in *ms* in both local and PMC execution modes for 30 workloads. Results are presented in three intensity levels of low, medium, and high beside mean of all intensity levels. Column $N$ in the Table represents the number of workloads whose execution times are used to calculate mean values of the minimum, maximum, and mean represented in the Table's columns. For instance, $N = 10$ shows that values shown in minimum, maximum, and mean columns are mean values of execution time for 10 workloads.

The results in the Table testify considerable achievement in reduction of execution time of RMA when running in PMC mode for all intensity workloads. In low, medium, and high intensity levels, the minimum time savings (shown in minimum column) are as high as 51%, 79%, and 87.5% which are increasing as the workload intensities raise. The maximum time saving for low, medium, and high intensity workloads are 67.3%, 82.3%,

Table 6.9: The execution time data generated via statistical modeling for local and PMC execution modes

| **Workload Number*** | **Intensity** | **Local Execution Time (ms)** | **PMC Execution Time (ms)** |
|---|---|---|---|
| 1 | Low | 4807.33 | 2356.92 |
| 2 | | 5402.25 | 2556.71 |
| 3 | | 6031.66 | 2649.29 |
| 4 | | 6593.59 | 2784.08 |
| 5 | | 7257.61 | 2948.5 |
| 6 | | 7895.57 | 3177.87 |
| 7 | | 8931.03 | 3303.69 |
| 8 | | 9762.63 | 3462.45 |
| 9 | | 10512.62 | 3570.17 |
| 10 | | 11288.5 | 3685.92 |
| 11 | Medium | 22391.05 | 4676.62 |
| 12 | | 23628.78 | 4781.32 |
| 13 | | 24896.37 | 5048.76 |
| 14 | | 26195.26 | 5025.89 |
| 15 | | 27525.81 | 5442.88 |
| 16 | | 28889.09 | 5316.82 |
| 17 | | 30285.73 | 5525.2 |
| 18 | | 31716.72 | 6065.57 |
| 19 | | 33182.8 | 5970.15 |
| 20 | | 34685.02 | 6113.49 |
| 21 | High | 45753.81 | 5712.5 |
| 22 | | 52247.41 | 6239.97 |
| 23 | | 59008.84 | 7195.76 |
| 24 | | 66050.71 | 7887.32 |
| 25 | | 73385.77 | 8562.54 |
| 26 | | 81026.95 | 8826.16 |
| 27 | | 88986.88 | 9557.51 |
| 28 | | 97278.32 | 10231.53 |
| 29 | | 105914 | 10543.58 |
| 30 | | 114906.4 | 11029.3 |

* Workloads values are presented in Table 5.1

Table 6.10: Descriptive Statistics of Execution Time Generated Using Statistical Analysis in ms

| Description | Intensity | N | Minimum | Maximum | Mean |
|---|---|---|---|---|---|
| Local Execution Time | | 10 | 4807.33 | 11288.5 | 7848.28 |
| PMC Execution Time | Low | 10 | 2356.92 | 3685.92 | 3049.56 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Time | | 10 | 22391.05 | 34685.02 | 28339.66 |
| PMC Execution Time | Medium | 10 | 4676.62 | 6113.49 | 5396.67 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Time | | 10 | 45753.81 | 114906.4 | 78455.9 |
| PMC Execution Time | High | 10 | 5712.50 | 11029.3 | 8578.61 |
| Valid N (listwise) | | 10 | | | |
| Local Execution Time | | 30 | 4807.33 | 114906.39 | 38214.6149 |
| PMC Execution Time | All | 30 | 2356.92 | 11029.30 | 5674.95 |
| Valid N (listwise) | | 30 | | | |

and 90% respectively. Theses numerical results indicate that achievements in higher workloads are considerably higher than lower workloads. For all the 30 workloads (mean of all three intensity levels), minimum, maximum, and mean execution time saved are as high as 51%, 90%, 85.1%. Hence, time saving achievements increase as the workloads rise.

According to the reported numerical performance gains, it is expected to observe much higher execution time in local execution mode in higher intensity levels. The mean execution time of workloads in local execution mode, for the low intensity category, is as much as 2.57 times more than PMC execution which is remarkably high despite of low intensity of the workloads. Similarly, for medium and high intensity levels, the mean execution time of workloads in the mobile devices is as high as 5.25 and 9.14 times more than PMC execution mode, respectively. The mean time saving for 30 workloads, as the last segment in the Table 6.10 shows, is observed as high as 6.73 times when execution is performed outside the mobile device. For instance, if the local execution time takes about 38214*ms*, leveraging the proposed framework can perform the task in 5674*ms* and save execution time by 32540*ms* in average which is significant.

Table 6.11: Results of Paired Samples T-Test for Analyzing the Significance of Execution Time Conservation in PMC Mode Compared to Local Mode.

| **Paired Sample Test** | | | **Paired Samples Correlation** |
|---|---|---|---|
| T | df | Sig | Sig (2-tailed) |
| 5.857 | 29 | 0.000 | 0.000 |

Although the results of descriptive statistics summarized in Table 6.10 demonstrated in Figure 6.9 advocate significant achievements in reducing the execution time of the application in PMC, we further extend our analysis by performing paired samples T-Test to statistically verify that the mean execution times in local and PMC modes are significantly different. The results of paired samples T-Test are presented in Table 6.11.

As the Table 6.11 summarizes, the values $t(29)=5.857, p<0.01$ show significant difference between mean local and PMC execution times. Positive t-value testifies that mean local execution time is larger than mean PMC execution time. Hence, the time saving using our proposed framework is significant compared to local execution mode, on average. Figure 6.9 represents graphical view of mean execution times for 30 workloads in three intensity levels calculated in both local and PMC execution modes. Each bar in this chart depicts mean value of 30 iterations for each workload (each workload executed 30 times). The blue diagonal stripped bars represent execution times of performing workloads in the mobile device whereas green plus-marked bars show the execution times of workloads when performed outside the mobile device in PMC mode. Because computing capability of mobile device is limited, growth in workload intensities has significant rise on execution time on local execution mode. So, the native execution of highly intensive workloads remarkably takes longer time that leads to application responsiveness degradation. However, growth in the workload's complexity has notably lower impacts on application execution time when performing outside the mobile device due to higher computing sources of our cloud of nearby mobile devices.

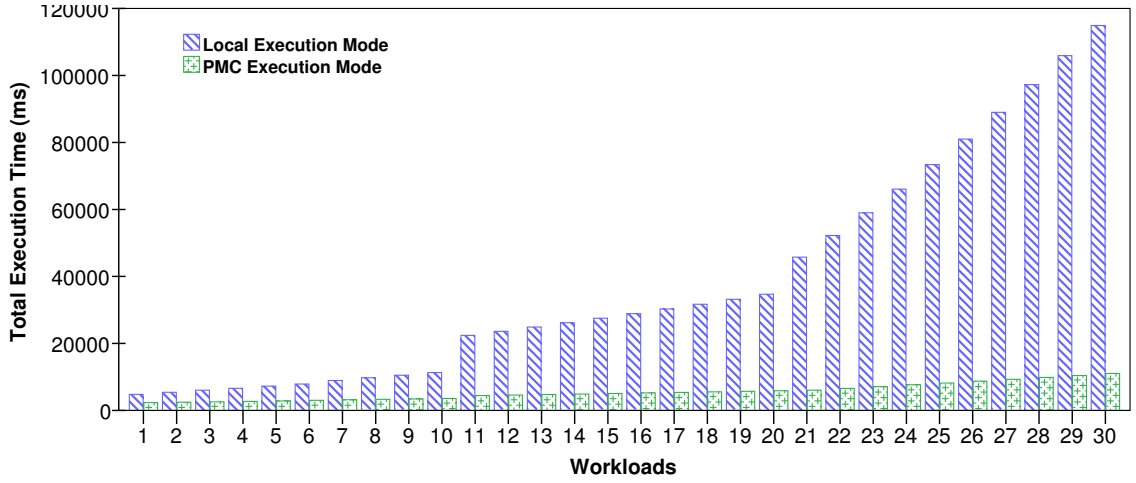



Figure 6.9: Execution Time for 30 Workloads Generated Via Statistical Analysis: Local vs PMC Execution Mode

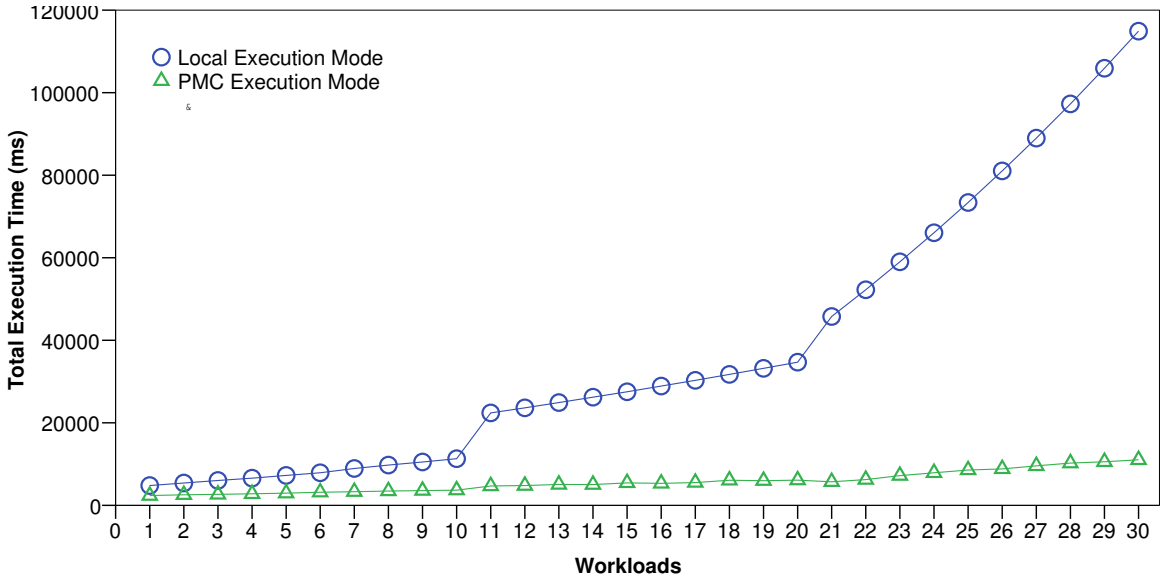



Figure 6.10: Scatter Plot with Interpolation Lines for Execution Time Generated Via Statistical Modeling

Figure 6.10 illustrates the achievement of time saving in our proposed framework. Scattered circles and triangles across the graph and corresponding interpolating lines unveil the differences in achievements and the correlation between the workload's intensity and time saving. In the first ten workloads with low intensity, the differences between circle and triangles are comparatively smaller than of medium and high intensity. The gap is increasing as the workload hikes. In the high workload intensity the difference is at the highest which advocates the relationship between the workload intensity and execution time.

The results in these Figures suggest that although PMC execution is always beneficial, the achievements are significant when execution complexity is high. Therefore, considering implications of wireless transmission (security, privacy, delay, and cost), utilizing PMC resources are discouraged for very low workloads. These results unveil that application responsiveness is notably improved and execution efficiency is remarkably enriched when leveraging the proposed MCC framework for execution of RMAs.

Execution time in local execution mode depends mostly on workload intensity and computing capabilities of the host mobile device (including CPU clock speed, RAM, storage, cache). However, the execution time in PMC execution mode depends on larger number of metrics, including computing and communication delay of discovering MSP (the time takes that MSC sends request and fetches the IP address(es) of MSP(s) from the TSG), wireless communication throughput, and computing power of MSP(s). Figure 6.11 shows the detailed timing of each contributing entity to the total execution time in PMC execution mode. Violaceous diagonal stripped chunks located in the bottom of the chart represent discovery delay of identifying potential MSPs in proximity. This delay includes

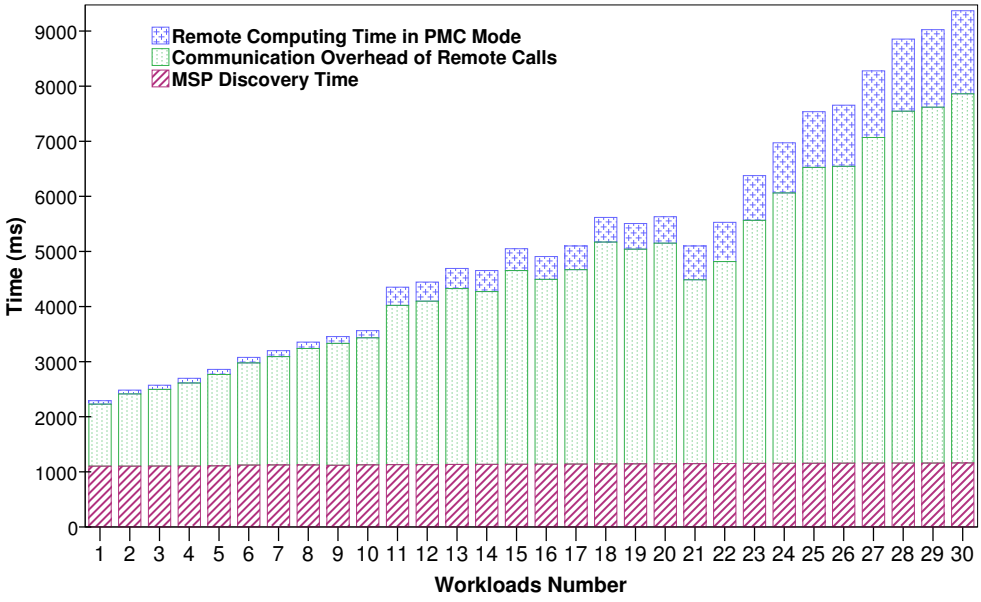



Figure 6.11: Breakdown of PMC Execution Time for 30 Workloads

the communication delay for the TSC to communicate with TSG to find nearby MSP and also searching time into the TSG database of MSPs.

Green dotted chunks show the communication latency when performing PMC execution calls and the blue marked chunks are PMC computing time. As the figure illustrates, the highest delays are communication overhead and discovery delay. However, these delays have negligible impact(s) on the user experience because the calls are performed asynchronously and hence user can unobtrusively continue interacting with the rest of application till the response arrives. During MSP discovery time, MSC sends asynchronous request to the TSG and ask for potential MSP(s) in vicinity. Upon successful discovery, the TSG sends the URI (IP addresses + resource ID) of services in the identified MSPs to the corresponding MSC. As the Figure 6.11 shows, the PMC execution time is prolonged mostly by wireless communication overhead. Such wireless communication delay of PMC calls has significant positive correlation with the amount of data transmitted over the entire execution process. So, the larger data is transmitted over the wireless network, the larger is the communication overhead.

Figure 6.12 illustrates the detailed information about size of data transmission in-

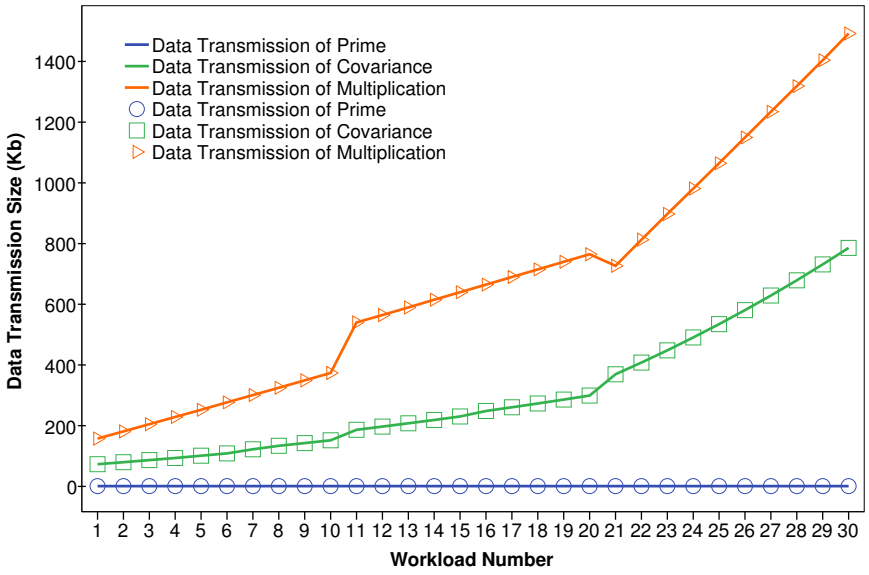



Figure 6.12: Interpolated Total Data Transmission in KB

cluding request and response size of each contributing service (i.e., prime, matrix multiplication, and matrix covariance) when communicating with the PMCs for 30 workloads. As the chart depicts, the workloads are selected to gradually increase data transmission size, so that the communication overhead in MCC is more effectively highlighted. The most data-intensive task is the matrix multiplication that is shown with orange triangles at the top of the chart and the least data-intensive service is prime that is shown by blue circles in the bottom of the diagram. Covariance is second data-intensive service that is shown with green squares in the middle of the chart. In the prime, all workloads are six-digit long from 200k to 900k. While the size of the workloads is the same, their numeric values significantly vary. So prime is computing-intensive rather than data-intensive or communication-intensive.

The marginal fluctuations (less than 10%) in the communication overhead shown in this chart follows the same pattern in Figure 6.11. This small fluctuation is due to slightly lower data transmission size for workload 21 and network inconsistency.

### 6.2.2 Consumed Energy

We have presented the consumed energy results in Table 6.12 for 30 workloads in three intensity levels of low, medium, and high collected in local and PMC execution modes. The results highlight remarkable energy saving when performing resource-intensive task in PMC execution mode. Table 6.13 presents descriptive statistics of the energy consumption of mobile device in *mJ* for executing workloads in both local and PMC execution modes. The results, including minimum, maximum, and mean values are presented for 30 workloads in three intensity levels of low, medium, and high beside mean of all intensity levels. Column *N* represents the number of workloads whose energy consumptions are used to calculate the mean values of minimum, maximum, and mean

represented in the Table's columns. For instance, $N = 10$ shows that values appeared in minimum, maximum, and mean columns are mean consumed energy values of 10 workloads. In low, medium, and high intensity levels, the minimum energy savings are as high as 26%, 63%, and 73% which are increasing as the workload intensities raise. The maximum time saving for low, medium, and high intensity workloads are approximately 46%, 68%, and 81% respectively. The mean energy saving of low, medium, and high intensity workloads is 39%, 66%, and 79% respectively.

Moreover, executing applications inside the mobile device for low, medium, and high intensity workloads consumes as high as 1.64, 2.9, and 4.7 times more energy than PMC execution which is remarkable achievement. Mean energy consumption for 30 workloads, as the last segment in the Table shows, is observed as high as 3.6 times more than energy consumption of the application when intensive tasks run outside the mobile device. For instance, if the energy required to locally run a workload is $12126mJ$, by utilizing the proposed framework in this study, the same workload consumes as low as $3370mJ$ energy by performing intensive task(s) outside the mobile device which is remarkable.

Similar to time analysis, we perform paired samples T-Test to statistically demonstrate significance of energy saving and RMA energy-efficiency using our proposal. The results of the paired samples T-Test are presented in Table 6.14.

As the Table 6.14 represents, the values $t(29)=5.623$, $p<0.01$ show significant differences between mean local and PMC energy consumption. Positive t-value testifies that mean local energy consumption is larger than mean PMC energy usage. Therefore, the energy saving using our proposed framework is significant compared to local execution mode, on average.

Figure 6.13 represent graphical view of energy consumption of 30 workloads in three intensity levels calculated in both local and PMC execution modes. Each purple bar repre-

Table 6.12: The consumed energy data generated via statistical modeling for local and PMC execution modes

| Workload Number* | Intensity | Local Consumed Energy (mJ) | PMC Consumed Energy (mJ) |
|---|---|---|---|
| 1 | Low | 1834.3 | 1356.41 |
| 2 | | 2075.59 | 1434.57 |
| 3 | | 2299.86 | 1512.65 |
| 4 | | 2463.23 | 1593.96 |
| 5 | | 2693.71 | 1684.33 |
| 6 | | 2901.17 | 1806.13 |
| 7 | | 3199.17 | 1905.35 |
| 8 | | 3532.77 | 1998.02 |
| 9 | | 3764.77 | 2062.72 |
| 10 | | 4016.91 | 2172.1 |
| 11 | Medium | 7313.66 | 2697.97 |
| 12 | | 7686.07 | 2791.52 |
| 13 | | 8080.06 | 2888.83 |
| 14 | | 8475 | 2986.5 |
| 15 | | 8813.63 | 3085.04 |
| 16 | | 9296.56 | 3200.51 |
| 17 | | 9769.23 | 3300.9 |
| 18 | | 10133.49 | 3402.55 |
| 19 | | 10661.04 | 3504.66 |
| 20 | | 11378.66 | 3608.17 |
| 21 | High | 13892.88 | 3705.81 |
| 22 | | 15834.22 | 4031.32 |
| 23 | | 18090.88 | 4358.68 |
| 24 | | 20504.53 | 4687.72 |
| 25 | | 22852.53 | 5019.01 |
| 26 | | 25104.89 | 5364.63 |
| 27 | | 28153.68 | 5707.86 |
| 28 | | 30182.95 | 6054.47 |
| 29 | | 32961.51 | 6407.3 |
| 30 | | 35829.17 | 6781.79 |

* Workloads values are presented in Table 5.1

Table 6.13: Descriptive Statistics of Consumed Energy Data Generated Using Statistical Analysis

| Execution Mode | Intensity | N | Minimum | Maximum | Mean |
|---|---|---|---|---|---|
| Local Execution | | 10 | 1834.30 | 4016.91 | 2878.14 |
| PMC Execution | Low | 10 | 1356.41 | 2172.10 | 1752.63 |
| Valid N (listwise) | | 10 | | | |
| Local Execution | | 10 | 7313.66 | 11378.66 | 9160.74 |
| PMC Execution | Medium | 10 | 2697.97 | 3608.17 | 3146.66 |
| Valid N (listwise) | | 10 | | | |
| Local Execution | | 10 | 13892.88 | 35829.17 | 24340.72 |
| PMC Execution | High | 10 | 3705.81 | 6781.79 | 5211.86 |
| Valid N (listwise) | | 10 | | | |
| Local Execution | | 30 | 1834.30 | 35829.17 | 12126.54 |
| PMC Execution | All | 30 | 1356.41 | 6781.79 | 3370.39 |
| Valid N (listwise) | | 30 | | | |

sents energy required to performing the one workload in the mobile device whereas each orange stripped bar shows the energy requirement of one workload when executed outside the mobile device.

Figure 6.14 is depicted to better highlight the achievement of this framework. Scat-

Table 6.14: Results of Paired Samples T-Test for Analyzing the Significance of Energy Conservation in PMC Mode Compared to Local Mode.

| Paired Sample Test | | | Paired Samples Correlation |
|---|---|---|---|
| T | df | Sig | Sig (2-tailed) |
| 5.623 | 29 | 0.000 | 0.000 |

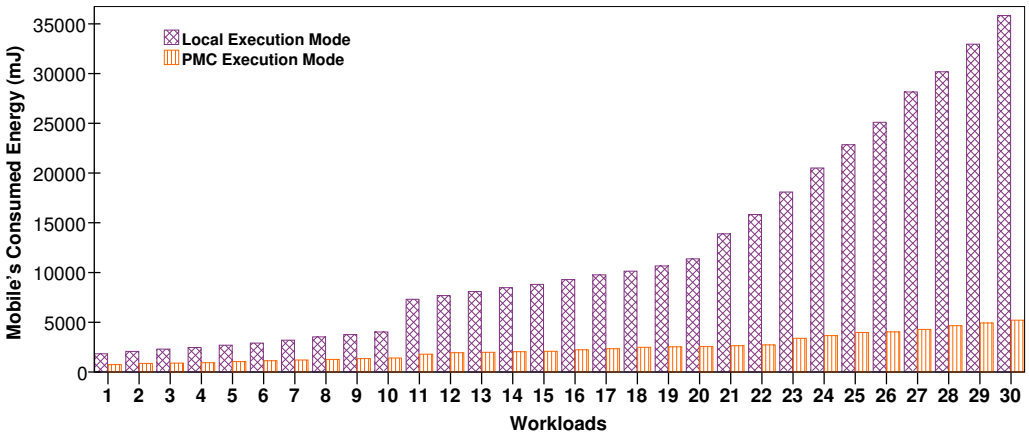



Figure 6.13: Consumed Energy for 30 Workloads Generated Via Statistical Analysis: Local vs PMC Execution Mode

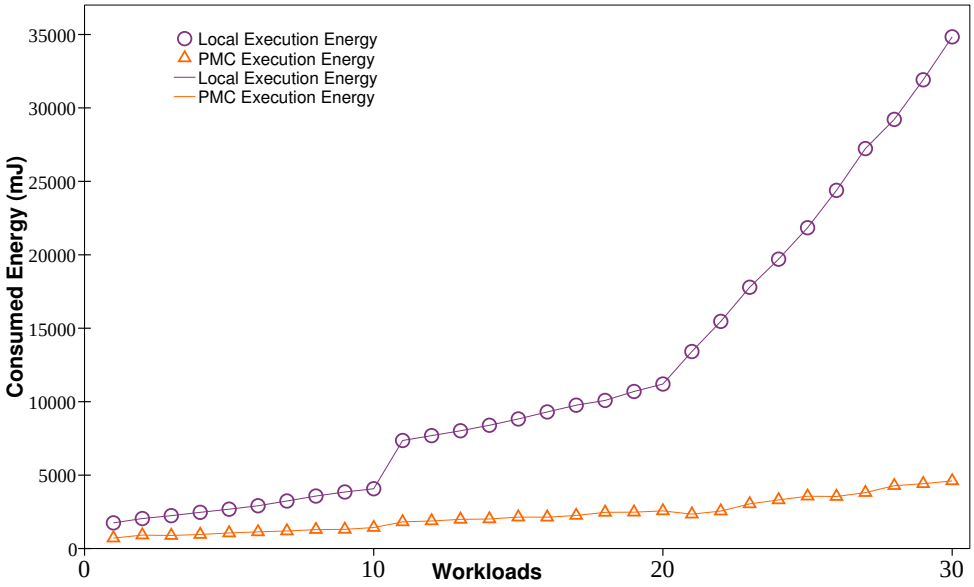



Figure 6.14: Scattered Plot with Interpolation Lines for Mobile Consumed Energy

tered circles and rectangles across the graph and corresponding interpolating lines show the differences in achievements and the correlation between the workload's intensity and energy saving. In the first ten workloads with low intensity, the difference between circle and rectangle is comparatively smaller than of medium and high intensity. The gap is increasing as the workload hikes. In the high workload intensity the difference is at the highest which advocates the relationship between the energy consumption and workload intensity.

Comparing and contrasting results appeared in Figures 6.13 and 6.14 suggest correlation between the execution time and energy consumption of the mobile device. There will be high probability that energy requirement has direct positive correlation with execution time; the longer is the execution time, the higher would be energy requirement. In order to investigate this correlation, we perform further analysis on execution time and consumed energy in both local and PMC execution. The results show remarkable positive correlation between the execution time and consumed energy in both execution modes. Figures 6.15 and 6.16 illustrate and highlight the correlations between execution time and energy consumption in local and PMC execution modes for 30 workloads. The $R^2$ values in these

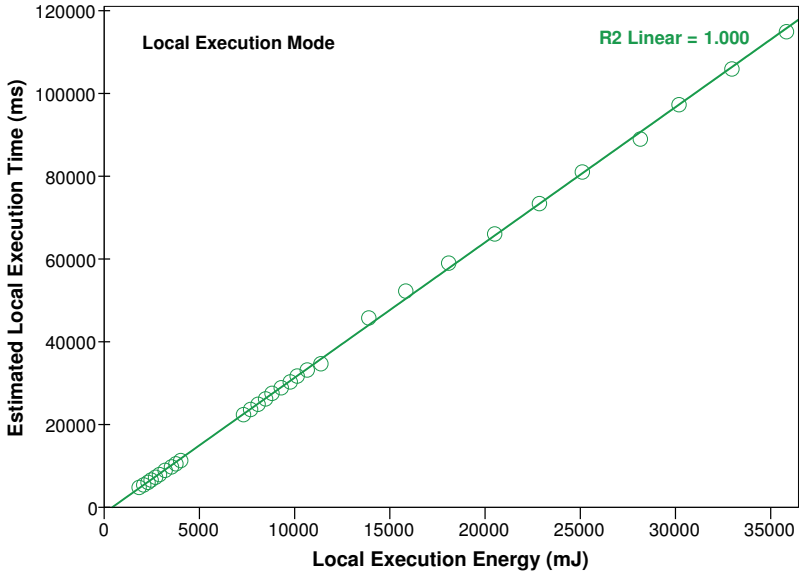



Figure 6.15: Linear Correlation Between Execution Time and Consumed Energy Generated via Statistical Analysis in Local Execution Mode

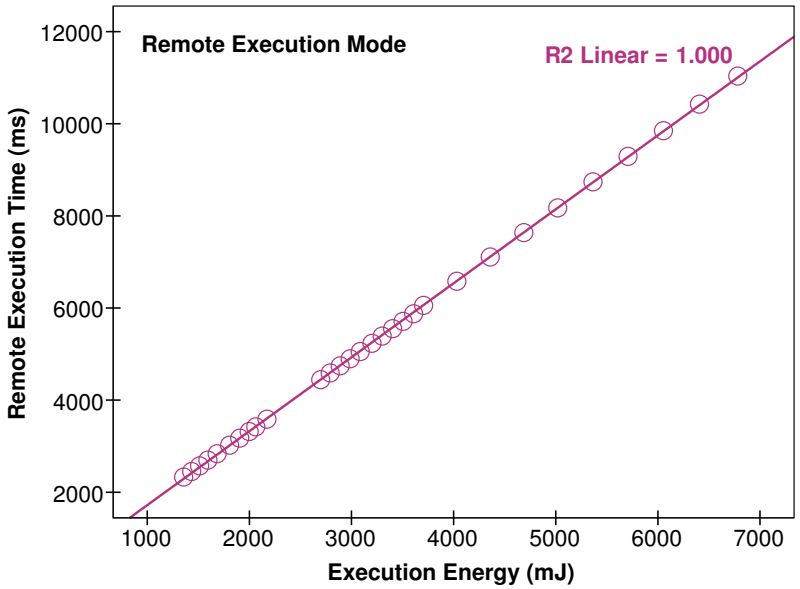



Figure 6.16: Linear Correlation Between Execution Time and Consumed Energy Generated via Statistical Analysis in PMC Execution Mode

charts indicate perfectly significant correlations between time and energy. Hence, increase in execution time of RMAs directly increases the energy consumption of the mobile device. It is noteworthy that in PMC execution mode, the excess communication overhead is manifested in PMC execution time. Therefore, the total energy is indirectly a functional of both execution time and wireless communication overhead.

Thus, in local execution mode, due to limitations of mobile device, growth in work-

load intensities has significant rise on execution time and consequently on energy requirement of the mobile device. So, execution of highly intensive values remarkably takes more energy that cause higher energy dissipation. However, in PMC execution mode, growth in workloads' complexity has minimal impacts on application energy consumption because of higher PMC resources and computing capabilities.

The results in Figures 6.13 and 6.14 suggest that although PMC execution is always beneficial, the PMC execution has higher benefit on higher complexity workloads. Therefore, considering implications of PMC execution in wireless ecosystem, utilizing PMC resources are discouraged for very low workloads.

## 6.3 Platform-Independence Results

In this section, we present the results of our experiments described in Section 5.3. This experiment aims to demonstrate that the performance of our framework does not depend on the programming language, device, and benchmarks.

Results of our experiments using ASP and PHP are presented for 10 benchmarks with 99% confidence interval in Tables 6.15 and 6.16, respectively.

Descriptive statistics of analyzing execution time for local and PMC execution modes generated via ASP and PHP codes are summarized in Table 6.17. As the results demonstrate, there are significant performance gain when executing tasks using our framework

Table 6.15: Execution Time Generated Using ASP With 99% Confidence Interval

| Benchmarks | ASP Execution Time (ms) | | | | | |
|---|---|---|---|---|---|---|
| | Local | Error Estimate | Time with 99% Conf. Int. | PMC | Error Estimate | Time with 99% Conf. Int. |
| 9000 | 13520.45 | 62.01 | 13520.45(+/-)62.01 | 4770 | 17.99 | 4770.94(+/-)33.4 |
| 10800 | 19249.71 | 47.15 | 19249.71(+/-)47.15 | 6400 | 29.78 | 6399.62(+/-)55.28 |
| 12600 | 26190.37 | 62.66 | 26190.37(+/-)62.66 | 8350 | 60.17 | 8350.07(+/-)111.72 |
| 14400 | 34190.11 | 56.45 | 34190.12(+/-)56.45 | 10570 | 45.91 | 10570.09(+/-)85.23 |
| 16200 | 43549.89 | 99.88 | 43549.9(+/-)99.88 | 13009 | 29.16 | 13009.8(+/-)54.13 |
| 18000 | 53330.51 | 90.57 | 53330.52(+/-)90.57 | 15920 | 17.99 | 15919.94(+/-)33.4 |
| 19800 | 66000.93 | 110.42 | 66000.93(+/-)110.42 | 19050 | 55.21 | 19050.47(+/-)102.51 |
| 21600 | 77999.88 | 75.06 | 77999.88(+/-)75.06 | 22570 | 107.94 | 22569.63(+/-)200.41 |
| 23400 | 90000.15 | 106.08 | 90000.15(+/-)106.08 | 26180 | 75.06 | 26179.88(+/-)139.36 |
| 25200 | 102000.6 | 129.03 | 102000.69(+/-)129.03 | 30340 | 60.17 | 30340.07(+/-)111.72 |

Table 6.16: Execution Time Generated Using PHP With 99% Confidence Interval

| Benchmarks | PHP Execution Time (ms) | | | | | |
|---|---|---|---|---|---|---|
| | Local | Error Estimate | Time with 99% Conf. Int. | PMC | Error Estimate | Time with 99% Conf. Int. |
| 9000 | 3990 | 15.51 | 3990.64(+/-)15.51 | 1839 | 25.43 | 1831.84(+/-)25.43 |
| 10800 | 5350 | 26.06 | 5349.67(+/-)26.06 | 2201 | 8.06 | 2200.73(+/-)8.06 |
| 12600 | 6930 | 44.05 | 6929.61(+/-)44.05 | 2618 | 17.99 | 2617.94(+/-)17.99 |
| 14400 | 8640 | 30.4 | 8640.45(+/-)30.4 | 3006 | 25.43 | 3004.84(+/-)25.43 |
| 16200 | 11150 | 55.21 | 11750.47(+/-)55.21 | 3591 | 17.99 | 3590.939(+/-)17.99 |
| 18000 | 12910 | 31.64 | 12910.1(+/-)31.64 | 4248 | 37.84 | 4248.352(+/-)37.84 |
| 19800 | 15480 | 56.45 | 15480.12(+/-)56.45 | 4870 | 26.06 | 4870.669(+/-)26.06 |
| 21600 | 18180 | 35.98 | 18180.38(+/-)35.98 | 5581 | 24.19 | 5581.192(+/-)24.19 |
| 23400 | 21170 | 55.21 | 21170.47(+/-)55.21 | 6470 | 44.67 | 6470.43(+/-)44.67 |
| 25200 | 24360 | 30.4 | 24360.45(+/-)30.4 | 7390 | 32.88 | 7389.74(+/-)32.88 |

in both programming languages. When running our framework using ASP code, the minimum, maximum and mean time saving of using PMC mode are 64.7%, 70%, and 70%, respectively.

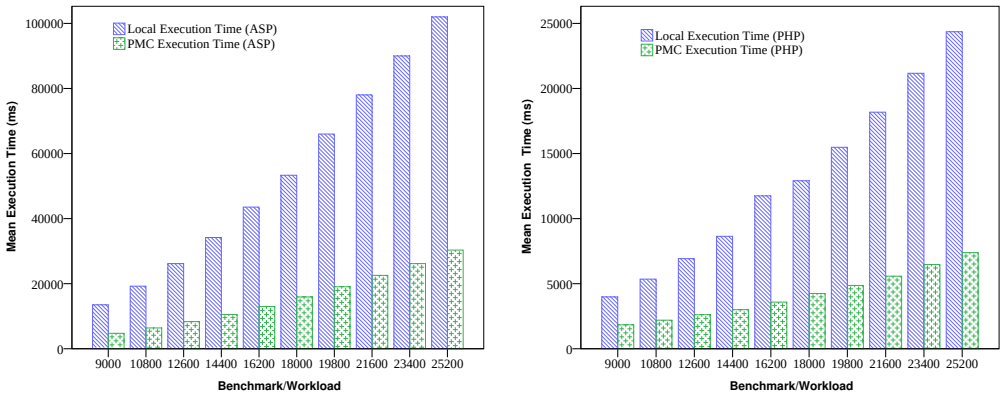
When performing execution of the PHP code, the results testify highly similar results. The minimum, maximum, and mean time saving of running the tasks using our framework are 46%, 69.6%, and 67.5%. These findings testify that lightweight feature of our framework for execution of compute-intensive tasks is not depend on the programming language, mobile device, and benchmark values.

We plot bar charts and scatter diagrams to portray our results and findings in Figures 6.17 and 6.18. Bar charts 6.17(a) and 6.17(b) present the results of our benchmarking experiments written in ASP and PHP for 10 benchmarks/workloads in two execution modes of local and PMC (remote execution using our framework).

The diagonally strapped bars in both chars show local mean execution times, whereas

Table 6.17: Descriptive Statistics of Execution Time in Local and PMC Mode via ASP and PHP

| Description | N | Minimum | Maximum | Mean |
|---|---|---|---|---|
| Local Execution Time (ASP) | 10 | 13520.45 | 102000.7 | 52603.27 |
| PMC Execution Time (ASP) | 10 | 4770.94 | 30340.07 | 15716.05 |
| Valid N (listwise) | 10 | | | |
| Local Execution Time (PHP) | 10 | 3990.64 | 24360.45 | 12876.24 |
| PMC Execution Time (PHP) | 10 | 1831.84 | 7389.75 | 4180.668 |
| Valid N (listwise) | 10 | | | |



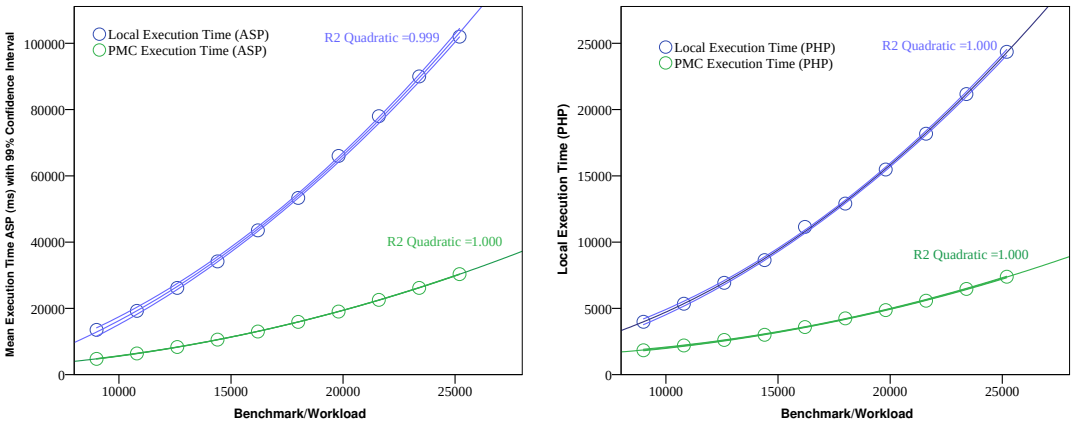

(a) Execution Times Generated via ASP Code (b) Execution Times Generated via PHP Code

Figure 6.17: Execution Times for 10 Benchmarks Generated Via Benchmarking: Local vs PMC Execution Modes

the patterned bars indicate the PMC mean execution time. Visual comparison of the results highlight the significance of our achievement using the proposed framework. Although execution of tasks takes longer in ASP compared to PHP, the performance gain is still almost the same.

As the results suggest, there is significant improvement over performing compute-intensive tasks such as Sieve using our proposed framework which is an evidence that out framework is lightweight and its performance is independent from programming language, execution platform, mobile device type, and benchmark type. As the workloads are increasing from left to right, the performance gains are also increasing. The larger benchmark the more performance gain.

We further analyzed our findings whose results are depicted in Figure 6.18. The scatter diagrams 6.18(a) and 6.18(b) show the mean execution time values for our 10 benchmarks/workloads with the best fit lines for local and PMC execution modes with 99% confidence interval . Green circular marks in the lower hapf of charts indicate time for performing task locally in the mobile device, whereas blue circular marks in the top half of the figures show the execution time in PMC execution mode when task is performed using our framework.




(a) Interpolated Execution Time Results Generated via ASP Code

(b) Interpolated Execution Time Results Generated via PHP Code

Figure 6.18: Scatter Plot With Fit Lines of Execution Times for 10 Benchmarks Generated Via Benchmarking: Local vs PMC Execution Modes

The best fit lines demonstrate quadratic correlations between the benchmark values and respected mean execution times that testify the fact and verifies that data generation is neither results of type-1 error nor are generated by chance. The correlation values of $R^2 = 1.00$ and $R^2 = 0.999$ in local and PMC modes for both ASP and PHP languages show that almost 100 % of the time values are statistically explainable by the benchmark values which follow the quadratic complexity of Sieve algorithm (Jain, 2008).

Comparison of the results of local vs. PMC execution mode generated using ASP and PHP languages, unveils that our framework is lightweight regardless of programming language, benchmarks, and mobile devices used in the experiments.

## 6.4 Comparative Study

The results of our comparative study are presented in this section. The results in this experiment demonstrate that the performance of our framework is comparatively more efficient that of offloading in Cloudlet.

The results in Table 6.18 shows the mean execution time of 10 benchmarks with 99% confidence interval in two execution modes of “Code Calling” and “Code Offloading”. The former refers to our framework whereas the latter is the execution mode following

Table 6.18: Execution Time for 10 Benchmarks in Two Execution Modes with 99% Confidence Interval

| Benchmarks | Code Calling | Error Estimate | Time with 99% Conf. Int. | Code Offloading | Error Estimate | Time with 99% Conf. Int. |
|---|---|---|---|---|---|---|
| 9000 | 1831.84 | 25.43 | 1831.84(+/-)25.43 | 23416.55 | 558.56 | 23416.55(+/-)558.56 |
| 10800 | 2200.73 | 8.06 | 2200.73(+/-)8.06 | 24206.53 | 473.7 | 24206.53(+/-)473.7 |
| 12600 | 2617.94 | 17.99 | 2617.94(+/-)17.99 | 24778.19 | 604.54 | 24778.19(+/-)604.54 |
| 14400 | 3004.84 | 25.43 | 3004.84(+/-)25.43 | 25151.62 | 434.36 | 25151.62(+/-)434.36 |
| 16200 | 3590.94 | 17.99 | 3590.94(+/-)17.99 | 25718.03 | 276.19 | 25718.03(+/-)276.19 |
| 18000 | 4248.35 | 37.84 | 4248.35(+/-)37.84 | 26425.26 | 358.73 | 26425.26(+/-)358.73 |
| 19800 | 4870.67 | 26.06 | 4870.67(+/-)26.06 | 27191.38 | 399.15 | 27191.38(+/-)399.15 |
| 21600 | 5581.19 | 24.19 | 5581.19(+/-)24.19 | 27939.72 | 498.71 | 27939.72(+/-)498.71 |
| 23400 | 6470.43 | 44.67 | 6470.43(+/-)44.67 | 28596.77 | 327.41 | 28596.77(+/-)327.41 |
| 25200 | 7389.75 | 32.88 | 7389.75(+/-)32.88 | 29150.34 | 693.46 | 29150.34(+/-)693.46 |

the code offloading in Cloudlet. As the result show there is a significant difference in execution time between using our framework and other frameworks that offload codes to the remote server.

The main reason for such a difference in execution time is the long WAN latency of transmitting run-time environment from mobile to the cloud using wireless communication technology. The result does not include the computing overhead of loading the run-time environment in the mobile side. If considering that overhead, the execution time results will be much more higher than current state.

Descriptive statistics of the results are presented in Table 6.19. The minimum, maximum and mean time saving values are as significant as 92.1%, 74.64%, and 84%, respectively.

We also performed paired sample t-test to demonstrate that execution time values reported in this experiment are significantly correlated by the execution modes. Table 6.20 summarized the results of the paired sample t-test. The $t = 292.43$ and $p < 0.01$ demonstrate that there is a statistically significant difference between execution time val-

Table 6.19: Descriptive Statistics of Execution Time in Local and PMC Mode via ASP and PHP

| Execution Mode | N | Minimum | Maximum | Mean |
|---|---|---|---|---|
| Code Calling | 10 | 23416.55 | 29150.34 | 26257.44 |
| Code Offloading | 10 | 1831.84 | 7389.75 | 4180.67 |
| Valid N (listwise) | 10 | | | |

ues reported in both execution modes. The positive $t$ value shows that the execution times via code calling are less than offloading mode.

We also performed Pearson correlation analysis to identify the correlations between the execution times and execution modes. We found Pearson correlation value of 98.7% with $p < 0.01(2-tailed)$ which advocates that there is a statistically significant difference between the execution modes and execution time values reported. We also performed Pearson correlation analysis to identify the correlation between the benchmark values and execution times ine ach execution modes. We found Pearson correlations of 98.9% with $p < 0.01(2-tailed)$ and 99.8% with $p < 0.01(2-tailed)$ between results and benchmark values for code calling and offloading, respectively. The results advocate statistically significant correlations between the reported results for benchmark execution time and benchmark value and suggests that as the benchmark values increase, the execution time is also increase with significant direct correlation.

Figure 6.19 depicts the mean execution time of 10 benchmarks in two execution modes. The patterned bars in the bottom of the chart depict the mean execution time for each benchmark that are generated using our framework. The diagonal stripped bars represent mean execution time when the run-time environment is offloaded along with the user data and code to the remote server. As the figure depicts there is significant difference between execution times in these two execution modes.

The findings are not limited to selected benchmarks only as it is evident in Figure 6.20. Interpolation lines in this figure depict interpolated execution time values for bench-

Table 6.20: Results of Paired Samples T-Test for Analyzing the Significance of Execution Time Difference in Two Execution Modes

| **Paired Sample Test** | | | **Paired Samples Correlation** |
|---|---|---|---|
| T | df | Sig | Sig (2-tailed) |
| 292.43 | 9 | 0.000 | 0.000 |

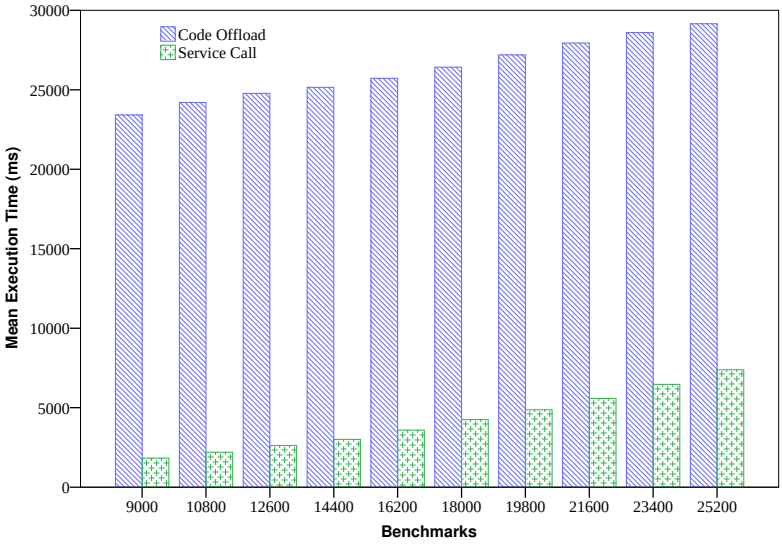



Figure 6.19: Execution Times of 10 Benchmarks Collected via Benchmarking in Two Execution Modes

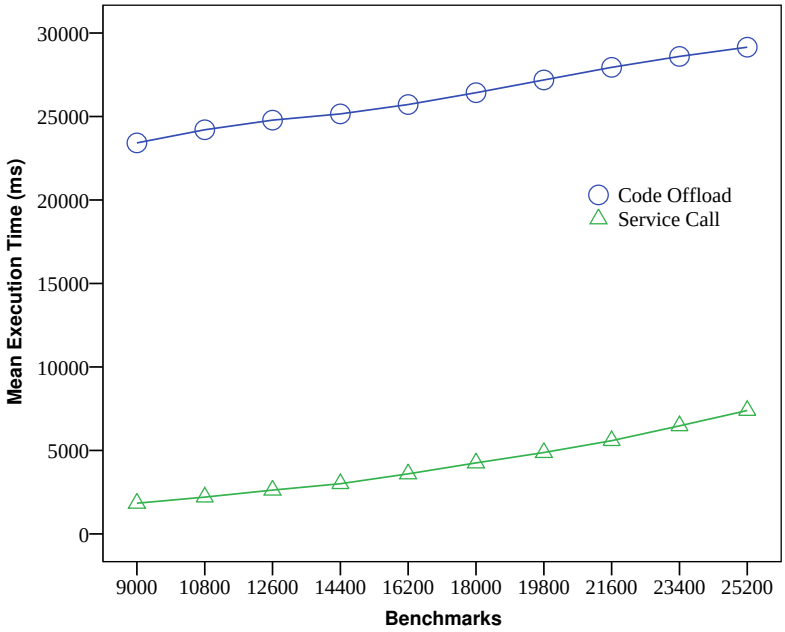



Figure 6.20: Interpolated Execution Times Results Collected via Benchmarking in Two Execution Modes

marks in the given benchmark range. Each mark represent the exact value of mean execution time for the specific benchmarks.

In this comparative study, we performed new set of experiments to compare the performance of the remote application execution mechanism of our framework with other related works. As the numerical, descriptive, and illustrative results demonstrate, leveraging remote application execution mechanism that is being proposed in our framework is significantly improving the application execution time in cloud-connected compute-intensive

mobile applications. Using this mechanism we could completely omit the computing and communication overhead of transmitting the run-time environment data from mobile to the remote resources.

In MCC augmenting computational capabilities of mobile devices has adverse correlation with the data transfer volume (Kumar & Lu, 2010) meaning that as the data transfer volume increase, the augmentation level decreases. Therefore, in design and development of MCC frameworks the data transfer needs to be mitigated to avoid excess computation (including application analysis, identifying intensive components, partitioning application, synchronizing memory state, and reintegrating results from remote servers) and communication (long WAN latency of transporting voluminous amounts of data and code from mobile to cloud) latencies.

The voluminous data transfer in MCC solution not only imposes long WAN latency, but also complicates system behavior prediction which is critical in MCC. Predicting the remote execution time of RMA in MCC is vital to adapt the application to different execution modes to fulfill user requirements and expectations. Due to bursty and unpredictable nature of wireless communication in MCC, excess data transmission increases the anomaly and make prediction harder.

The findings of this experiment advocate execution efficiency and lightweight characteristics of our proposed framework.

## 6.5 Validity Threats

Four types of validity threats for every computer system experimentation is identified by Cook et al (Cook, Campbell, & Day, 1979) that are considered in all experiments undertaken as part of performance evaluation methodology in this thesis. These four validity threats are described below.

### 6.5.1 Conclusion Validity

In order to build valid statistical model, we selected minimum of 30 workloads that are executed 30 times and reported using 99% confidence interval. We also used different statistical tools to avoid conclusion threats. We also performed a separate set of experiments using different mobile devices, programming languages, and even benchmarks to demonstrate that our findings are not biased. We are aware that our framework can yield varied degree of performance gain depending on the application type, mobile device, and programming language. Moreover, we experimentally compared remote application execution mechanism in our framework by other works in the literature to demonstrate its lightweight feature and execution efficiency. Thought, we could demonstrate that our framework is lightweight, we do not consider the results and our findings as final.

### 6.5.2 External Validity

We performed our experimentation over real test-bed and validated our performance evaluation results via statistical modeling. Our prototype applications represent prime and matrix algebra operations which are widely used in real word. We have used varied benchmark intensities so that benchmarks consume varied amounts of computation resources from the MSPs (Jain, 2008). We also used Sieve of Eratosthenes benchmarks which is a popular and standard benchmark (Jain, 2008) to demonstrate the performance gain of our framework. We know that varied types of benchmark might demonstrate varied performance results and findings. Thus, generalizing the results of this study is subject of the domain we apply the algorithms used in this thesis.

Moreover, we performed further experiments using different mobile phones, operating systems, programming languages, wireless access points, and computers to further generalize these results and findings. However, further generalization is possible and need

further experiments using other mobile devices, programming languages and benchmarks.

### 6.5.3 Internal

This study is closely monitored and controlled, particularly the impact of external entities that can influence our data collection accuracy are considered at the time of data collection. In particular, we closely monitored the execution characteristics of the mobile device in the presence of bursty networking. To address the impact of networking, we performed data collection after peak office hours after 6 PM local time throughout the study. We also monitored energy consumption of the mobile device closely and observed that when device's battery level is less than 50 %, the energy data collected via our instrument shows significant anomaly that degrades validity of our data collection. Throughout the study, we avoid data collection when battery level is less than 50%.

Moreover, the time data collector instrument is fully automatic to avoid man-made mistake. We made several pilot study prior main data collection to make sure the built-in timers are collecting data adequately. For energy data collection, we used a software-based instrument that is highly credible in the literature and can collect reliable energy data throughout the experiments. For adequate collection of data from this instrument, we used regular expression and automatically extracted the results under supervision of experimenter to ensure that no man-made mistake is encountered.

### 6.5.4 Construct

We have carefully designed our experimentations and piloted them on several occasions before final data collection. One of the threats we could omit from our validity is that the Samsung Galaxy S2 smartphone that was selected for experiment was not appropriate and energy data collection tools could not collect proper energy data, because it is unable to collect energy data related to the Wi-Fi radio transmission. Hence, we utilized another

smartphone (HTC Nexus One) whose energy consumption of processing and Wi-Fi elements can be precisely measured using our selected instrument.

## 6.6 Discussions

Findings of our evaluation via benchmarking experiments and validation through statistical analysis are separately reported in sections 6.1 and 6.2 in this chapter. In this section, we briefly present the synthesis results of the findings of benchmarking and statistical analyses results to validate the findings of this research. The discussions are presented in two sections of execution time and consumed energy.

### 6.6.1 Execution Time

For execution time, we synthesized and compared the results of benchmarking analysis and statistical modeling. Table 6.21 presents comparison among descriptive statistics of execution time results of benchmarking and statistical modeling methods. As the Table shows, results of benchmarking and statistical modeling advocate significant time saving in leveraging our framework.

The minimum and maximum time savings of exploiting our framework are as significant as 62.30% and 91.40%, respectively when used benchmarking analysis. Similarly, the minimum and maximum time savings when leveraging PMC execution mode are reported as high as 50% and 90.4% using statistical model. The mean time saving for local and PMC execution modes is 85.14% via statistical analysis and 87% via benchmarking results. Such proximity and strong support of findings advocate reliability and validity of the execution time analysis. Figure 6.21 illustrates mean of execution time generated using benchmarking and statistical analysis in local and PMC execution mode that signifies the difference between the execution time in local and PMC execution modes.

To demonstrate the reliability of data collected in benchmarking and statistical mod-

Table 6.21: Evaluation Methods Comparison: Execution Time in Local and PMC Execution Modes

| Evaluation Method | Local Execution Time (ms) | | PMC Execution Time(ms) | | Time Saving | |
|---|---|---|---|---|---|---|
| | | | | | Numeric (ms) | Percentage |
| Statistical Modeling | Min | 4807.33 | Min | 2356.92 | 2447.41 | 50.90% |
| | Max | 114906.4 | Max | 11029.3 | 103877.1 | 90.40% |
| | Mean | 38214.61 | Mean | 5674.95 | 32539.66 | 85.14% |
| Benchmarking | Min | 6102.2 | Min | 2298.6 | 3803.6 | 62.30% |
| | Max | 127080.4 | Max | 10939.17 | 116141.2 | 91.4% |
| | Mean | 42729.39 | Mean | 5526.67 | 37202.72 | 87% |

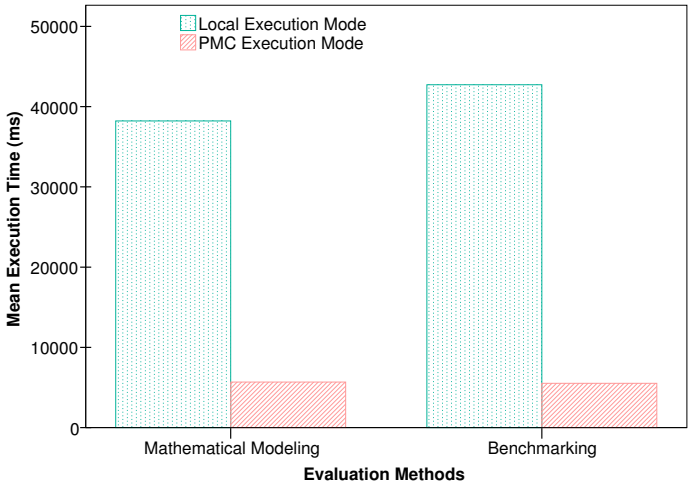
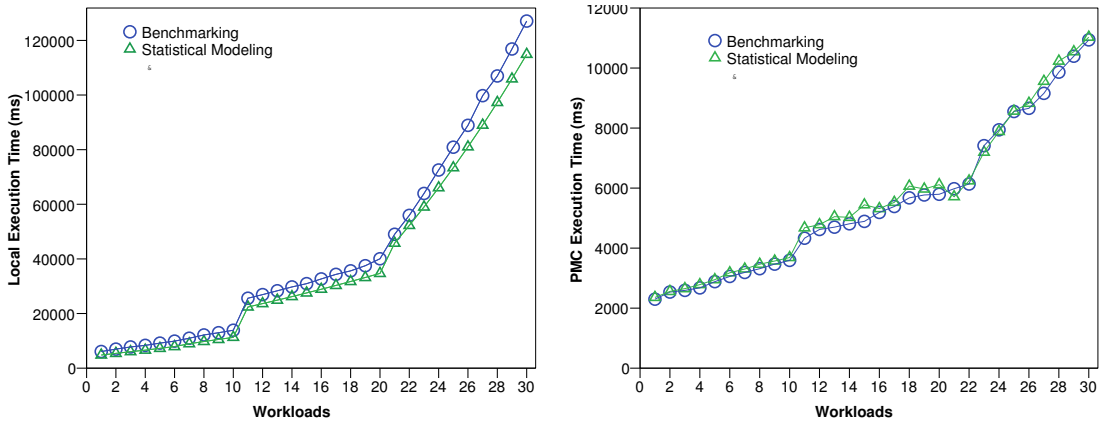



Figure 6.21: Comparison of Local and PMC Execution Time Data Collected via Statistical Analysis and Benchmarking

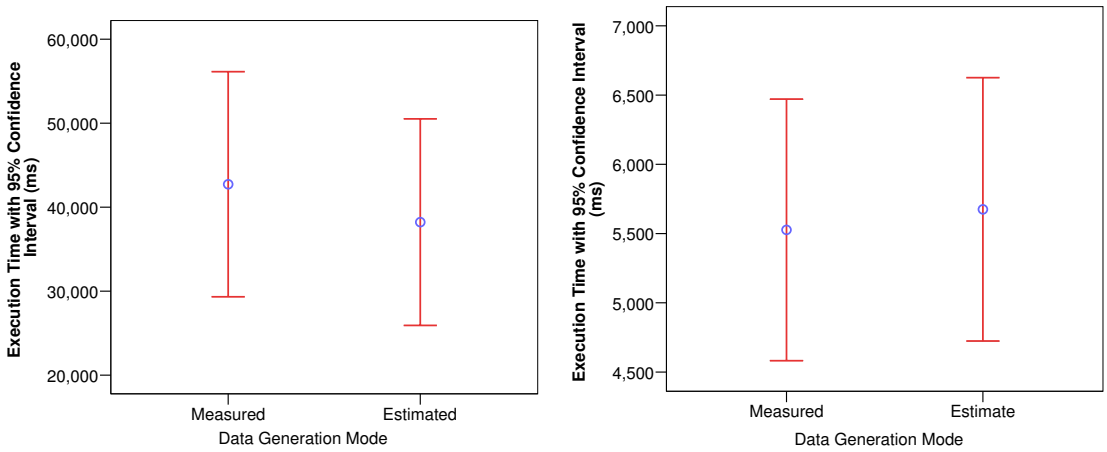
eling, we synthesize the results. The results of the synthesis are illustrated in Figure 6.22. The small error between data collected via benchmarking and statistical model advocate reliability and validity of our performance evaluation. Chart 6.22(a) shows the local execution time data generated via benchmarking and statistical modeling, and chart 6.22(b) demonstrates the PMC execution time data generated via benchmarking and statistical modeling. The results are closely supporting each other with small estimate error. The slight difference in execution time result of benchmarking and statistical modeling is due to implications and hard-to-predict feature of the real mobile environment.

We further plotted the error bars of local and PMC execution time data collected via benchmarking and statistical modeling with 95 % confidence interval which are presented




(a) Local Execution Time (b) PMC Execution Time

Figure 6.22: Comparison of Benchmarking and Statistical Results




(a) Local Execution Time (b) PMC Execution Time

Figure 6.23: Comparison of Benchmarking and Statistical Results With 99 % Confidence Interval

in Figures 6.23. The red lines in figures show the range of execution time values with 95% confidence interval and the blue circular remarks indicate the mean values of the corresponding data. The red lines and the blue circular remarks in charts 6.23(a) and 6.23(b) advocate the accuracy of the execution times generated in our performance evaluation for local and PMC modes. According to the rule of thumb, the overlapping lines indicate non-significant differences in values and also show that p value is less than 0.05. Hence the results of our evaluation are valid.

Table 6.22: Evaluation Methods Comparison: Energy Consumed in Local and PMC Execution Modes

| Analysis Method | Local Consumed Energy (mJ) | | PMC Consumed Energy (mJ) | | Energy Saving | |
|---|---|---|---|---|---|---|
| | | | | | Numeric | Percentage |
| Statistical Modeling | Min | 1834.3 | Min | 1356.41 | 477.89 | 26% |
| | Max | 35829.17 | Max | 6781.79 | 29047.38 | 81% |
| | Mean | 12126.54 | Mean | 3370.39 | 8756.15 | 72.20% |
| Benchmarking | Min | 1747.9 | Min | 1301.7 | 446.2 | 26.50% |
| | Max | 34838.2 | Max | 6585.5 | 28252.7 | 81% |
| | Mean | 11871.16 | Mean | 3388.65 | 8482.51 | 71.45% |

### 6.6.2 Consumed Energy

Similar to time comparison, we compare energy saving reported by benchmarking and statistical analysis methods in Table 6.22. As the descriptive statistics in the Table show, least and most energy savings percentage when leveraging PMC execution mode are as high as 26% and 81% using statistical model. The least and most energy savings percentage of exploiting our framework are as significant as 26.50% and 81% respectively in the benchmarking analysis. The mean energy saving for local and PMC execution modes is 72.20% via statistical analysis and 71.45% via benchmarking results. Figure 6.24 illustrates mean of consumed energy data generated using benchmarking and statistical analysis in local and PMC execution mode that signifies the difference between the energy consumption in local vs PMC execution modes.

In energy part, we compared the results of benchmarking analysis and statistical modeling. The results of the synthesis are illustrated in Figure 6.25. The small error between data collected via benchmarking and statistical model advocate reliability and validity of our performance evaluation. Chart 6.25(a) shows the local consumed energy data generated via benchmarking and statistical modeling, and chart 6.25(b) demonstrates the PMC consumed energy data generated via benchmarking and statistical modeling. The results are closely supporting each other with small estimate error.

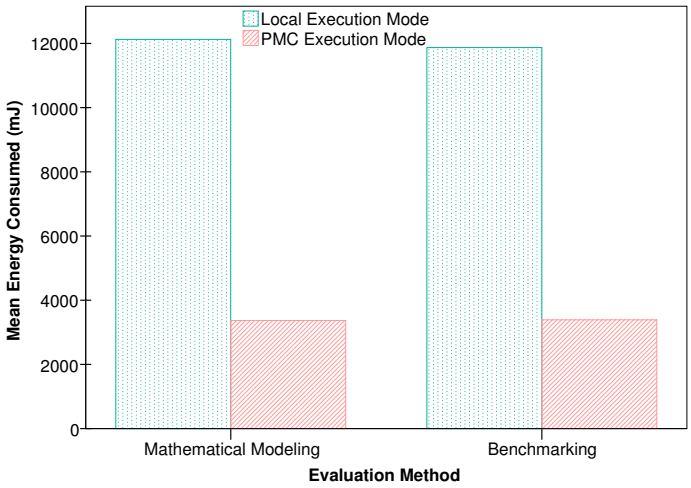
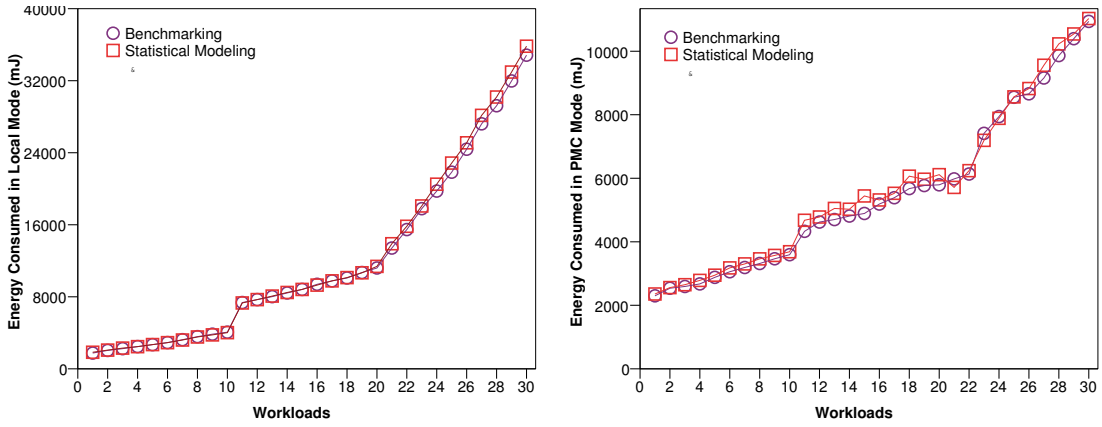


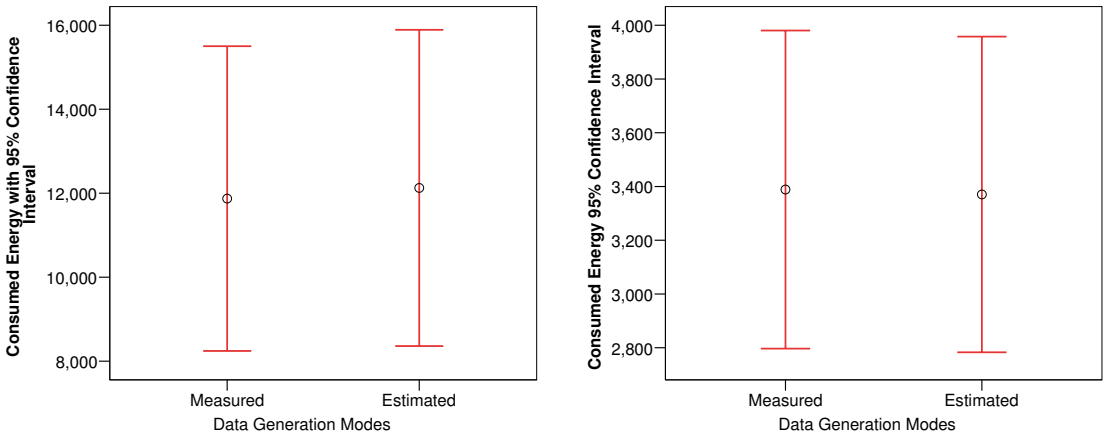

Figure 6.24: Comparison of Local and PMC Consumed Energy Data Collected via Statistical Analysis and Benchmarking




(a) Local Consumed Energy (b) Remote Consumed Energy

Figure 6.25: Comparison of Benchmarking and Statistical Results

We further plotted the error bars of local and PMC consumed energy data generated via benchmarking and statistical modeling with 95% confidence interval which are presented in Figures 6.26. The red lines in the diagrams show the range of consumed energy values with 95% confidence interval and the central circular remarks indicate the mean values of the corresponding data. The red lines and the central circular remarks in charts 6.26(a) and 6.26(b) advocate the accuracy of the energy data generated in our performance evaluation for local and PMC modes. According to the rule of thumb, the overlapping confidence interval lines indicate non-significant differences in values and also show that p-value is less than 0.05. Therefore, the results of our evaluation for energy are concluded




(a) Error Bars in Local Consumed Energy (b) Error Bars in PMC Consumed Energy

Figure 6.26: Comparison of Benchmarking and Statistical Results With 99 % Confidence Interval

to be valid.

## 6.7 Conclusions

In this chapter, the results of performance evaluation of our proposed framework via benchmarking and statistical analysis are reported and illustrated using several Tables and figures. The results are presented in sections 6.1 and 6.2 are synthesized in the section 6.6. Using paired samples t-test, we demonstrated significant time- and energy-efficiency yield from benchmarking and statistical modeling when performing intensive components of RMAs on PMC resources. The performance evaluation of the framework is carried out using workloads of three intensity levels to effectively highlight the correlation between workloads and time saving as well as energy saving. Although the proposed solution is remarkably effective in all intensity levels, the findings are more significant when the workloads are highly intensive. Time- and energy-efficiency rates are increasing as the workload intensities are increased.

The evaluation results testified about 86% time-efficiency as well as 72% energy-efficiency when the execution of intensive RMAs is performed outside the MSC, inside the MSPs. The maximum time-efficiency is unveiled as high as 91% and energy-efficiency as

significant as 81%. The results of benchmarking and statistical modeling are synthesized to demonstrate insignificant differences between the reported achievements. The supportive results of real time experiment and statistical modeling unveil lightweight nature of the framework and its usability and successful adoption in real scenarios. Our secondary experiments testified that the lightweight feature of our proposed framework does not depend on the programming language, mobile device and its underlying operating system, and selected benchmarks.

# CHAPTER 7

# CONCLUSIONS AND FUTURE WORKS

In this chapter, we present conclusions on the research undertaken in this thesis and highlight the future works. We re-examine the aim and objectives of this research to ensure that they are realized through the work reported in this thesis. We present the contributions of this study and highlight the significance and novelty of the proposed framework. Moreover, we provide the list of international peer reviewed scholarly publications that are accepted for publication or already published from the efforts undertaken in this thesis. The chapter ends by discussion on future works.

The remainder of this chapter is as follows. Section 7.1 presents the efforts undertaken to fulfill the aim and objectives of this study. The contributions of the thesis are presented in section 7.2. Significance of the framework proposed in this research is highlighted in section 7.3 and the international scholarly publications produced from the efforts in this thesis are listed in section 7.4. Finally, section 7.5 presents and future works.

## 7.1 Aim and Objectives Achievement

In this research, we aimed to achieve efficient RMA execution via a lightweight MCC framework that minimizes the computing overhead and communication latency by establishing a proximate cloud of mobile devices under supervision of a trusted service governor. In the following, we verify the accomplishment of the thesis's aim by re-examining the fulfillment of the objectives of this study.

### 7.1.1 Study the research advancements of the domain to identify a significant research problem

We attain this objective by comprehensively reviewing RMAs and their execution approaches in MCC. RMAs are analyzed, their characteristics and requirements are identified, and several open challenges in execution and adoption of RMAs are highlighted. Moreover, the most credible cloud-based mobile augmentation approaches that aim to alleviate RMA's challenges have been critically reviewed to draw a clear picture of the state-of-the-art RMA augmentation approaches in MCC.

In fulfillment of this objective, we critically analyzed and synthesized cloud-based resources to devise the taxonomy of cloud-based computing resources. In this taxonomy, four types of cloud-based resources, namely (1) Distant Immobile Clouds, (2) Proximate Immobile Computing Entities, (3) Proximate Mobile Computing Entities, and (4) Hybrid resources are identified. These resources comprehensively are compared from different aspects, including proximity, availability, scalability, flexibility, utilization cost, and mobility. Furthermore, the state-of-the-art RMA empowerment efforts are critically studied considering the taxonomy of cloud-based resource to devise the taxonomy of approaches. Moreover, several credible efforts are compared and several open research challenges including long WAN latency, lack of lightweight CMA approaches, and high cost/overhead of mobile application distributed execution are highlighted considering their impact on QoS of RMA execution in MCC.

### 7.1.2 Investigate the identified research problem

Investigation on the identified problem of this thesis is successfully accomplished by performing analytical and empirical studies. The analytical analysis is undertaken to formulate the execution time and energy consumption of two RMAs in chapter 3. Empirical analysis is employed to validate the results and findings through benchmarking using

android-based smartphone and Amazon Cloud VM instances. In order to analyze execution time and energy consumption of two RMAs, namely prime and matrix product we ran the applications in three execution environments. Firstly, we execute applications locally inside the mobile device. Then we run the application on two different execution environments built using Amazon EC2 virtual machines located in Singapore and Sydney regions. The Singapore cloud is selected as the proximate resource to the mobile device, whereas the Sydney cloud is selected as a distant cloud to analyze the impact of distance between mobile and cloud on RMA's execution. The results of analytical and empirical analyses advocate that the number of intermediate hops and data transfer volume are the most important factors contributing to the long WAN latency between mobile and cloud resources. Processing delays in large number of intermediate hops scattered in long distance between mobile and cloud resources significantly increases application execution time and energy consumption. However, our analyses unveil that physical distance between mobile and client has negligible impact on time and energy, considering the high signal propagation speed (speed of light) in digital mediums.

### 7.1.3 Propose a lightweight MCC framework to enhance execution time and energy consumption of RMAs

We achieved this objective by proposing a lightweight MCC framework that harnesses the computing capabilities of cloud of Proximate Mobile Devices (PMCs). In order to mitigate the time and energy impacts of long WAN latency due to large number of intermediate hops between the mobile device and cloud-based resources, a lightweight framework is proposed based on ROA to perform resource-intensive parts of the RMAs using resources of the PMCs. Mobile devices that utilize computing services of our framework are called Mobile Service Consumer (MCS) and those lending their resources are called Mobile Service Provider (MSP). The main principle in design and development of this

framework is that building a proximate cloud consists of voluminous computing resources of numerous nearby MSPs can create a resourceful localized mobile datacenter to perform computing tasks of nearby MSCs. Proximity between the MSC and MSP significantly mitigates the impact of long WAN latency in blipful wireless network. Moreover, instead of using energy-hungry cellular networking technology, the energy-friendly WLAN can be utilized to save energy.

Utilizing ROA in this framework enables building loosely coupled service-based RMAs. Overhead of identifying, partitioning, and offloading the codes to remote resources in loosely coupled applications is significantly mitigated that leads to remarkable time and energy improvement in RMA execution. Hence, the lightweight feature of our proposed framework is attained and RMA execution can take place with remarkable time and energy saving.

### 7.1.4 Evaluate the proposed lightweight MCC framework

We successfully attain our objective by evaluating the performance of proposed MCC framework via series of benchmarking experiments in real MCC environment using android-based smartphones and Windows-based laptops via different programming languages. We performed several experiments for evaluation purpose. Firstly, we performed a series of experiments developed using PHP programming language and ran 30 workloads on an Android-based smartphone. Application execution time and consumed energy of the RMA in local and remote execution modes are measured, collected, analyzed, and synthesized to demonstrate the performance of the proposed framework. The results of this performance evaluation experiments unveiled more than 80 % time saving and more than 70% energy saving which are significant.

We further evaluated the lightweight feature of the proposed framework using new

series of experiments developed using ASP programming language over a windows-based mobile device to demonstrate that the performance of our framework does not depend on a certain programming language or mobile device.

The performance evaluation is further extended to a comparative study in which the mechanism employed in our framework to remotely execute the intensive components is compared with related works. This comparative study advocated the lightweight feature of our framework by showing that using our remote execution mechanism execution time of intensive tasks can be reduced to as high as 92%.

### 7.1.5 Validate the proposed lightweight MCC framework

The performance evaluation results are validated using statistical modeling. We use linear regression analysis as the most predominant observation-based modeling approach to derive the statistical models of execution time and energy consumption of the mobile application on local and PMC execution modes. We create a dataset using independent replication method to train the regression models and derive the statistical models. The statistical models are validated using split-sample approach and the results are used to verify the model's validity. We synthesize the results of time and energy generated via benchmarking and statistical modeling to validate performance evaluation results of our framework. The results advocate reliability and validity of the proposed framework and ensure that we achieved our aim.

Considering successful accomplishment of our objectives in this thesis, we conclude that the aim of this research is successfully realized.

## 7.2 Contributions

In this thesis, we have produced several contributions to the body of knowledge that are described as follows.

### 7.2.1 Taxonomy of Resource-intensive Mobile Application Development Challenges

In this thesis, we produced the first comprehensive survey of resource-intensive mobile applications that is published in the literature, as one of the most significant contributions. We have reviewed RMAs by critically reviewing 110 articles extracted from major scholarly digital libraries that are presenting the state-of-the-art researches to devise the taxonomy of the RMAs in MCC. Our study -briefly appeared in chapter 2- is the first published comprehensive survey and review of RMAs.

### 7.2.2 Taxonomy of Cloud-based Resources

Taxonomy of cloud-based resources is another contribution of this thesis yielded from literature review. In order to identify the major cloud-based computing resource and devise their taxonomy, we reviewed 185 articles and synthesize their features horizontally and vertically. The taxonomy of cloud-based computing resources, including distant immobile clouds, proximate immobile computing entities, proximate mobile entities, and hybrid resource is the first effort focused on characteristics and features of varied cloud-based resources. This work is presented in chapter 2 and published in the literature.

### 7.2.3 Impacts of Distant Immobile Clouds on RMA Execution in Cloud-based Mobile Empowerment

We contributed to the body of knowledge by identifying the impacts and implications of utilizing distant immobile clouds on efficient RMA execution in mobile empowerment solutions. We perform comprehensive analytical and benchmarking analyses on the long WAN latency of accessing distant cloud resources from mobile devices whose results are presented in chapter 3. We investigated and demonstrated significant overhead of utilizing distant cloud-based resource during cloud-based augmentation. The analytical model and real experiment revealed that physical distance between mobile and cloud has negli-

gible impact on the augmentation performance, whereas the number of intermediate hops and volume of data transfer have remarkable impacts on the efficacy of augmentation of resource-constraint mobile devices using cloud-based resources.

### 7.2.4 Lightweight MCC Framework

In this thesis, we presented one of the earliest works to leverage ROA in design and development of a lightweight MCC framework. Our proposed framework exploits computational powers of multitude of varied proximate mobile devices as ubiquitous MSPs and aims to achieve efficiency in RMA execution. This framework achieves its aim by asynchronously communicating with the nearby MSPs to remotely initiate execution of resource-intensive computations of RMAs in MSPs. Building a resource pool using numerous ubiquitous mobile devices creates an anywhere anytime computing cloud in vicinity that could remarkably enhance RMA execution's efficiency. The proposed lightweight MCC empowerment framework presented in chapter 4 -as one of the earliest efforts- deploys SOA and REST architectural style to build a low overhead cloud of nearby mobile devices. The proposed framework could address high computational and communicational overheads of accessing distant cloud-based resources, by providing a proximate mobile cloud that features high scalability and low latency. Using extensive analysis and experimentation we observed capabilities of contemporary mobile devices as ubiquitous computing service providers that can revolutionize the future of MCC.

### 7.2.5 Framework Evaluation and Validation

We contributed to the knowledge by implementing, evaluating, and validating the performance of our lightweight MCC framework to demonstrate its reliable and valid functionality, and significance. Detailed description of the systems and data that are used to evaluate and validate the proposal are reported in chapter 5 and the results of perfor-

mance evaluation and validation are presented in chapter 6. The results reveal feasibility and functionality of our proposed MCC framework. The results also verify that utilizing our proposed framework can achieve RMA execution efficiency and save execution time and energy as significant as 86% and 72% in average, respectively. The results explicitly ensure feasibility and lightweight characteristics of the framework and advocate that objectives of this study are fulfilled and the aim is realized.

### 7.3 Significance of the Work

Several key features -considered in design and development of this framework- that enhance the significance of this work are briefly discussed as follows.

- **Platform-independence:** Leveraging SOC in design and implementation of this framework makes this solution independent from varied mobile platforms. So, any mobile device including smartphone, laptop, tablet with any operating system including Android, iOS, Blackberry, and Windows mobile can leverage the benefits of this proposal with least deployment and configuration overheads.

- **Lightweight RMAs:** RMAs that are built according to this framework inherit lightweight attributes of ROA, particularly loose coupling of services with the rest of the application. Additionally, service-based RMA execution using this framework alleviates the overhead of exploiting virtualization technology. Augmenting execution of service-based RMA in MCC originates significantly lower overhead compared to tightly coupled codes in non-service-based applications. Enhancing performance of non-service-based applications obliges partitioning and separating the intensive codes from the application prior the remote execution, and also reintegrating the results after successful remote execution. Thus the RMAs deployed based on this framework feature the lightweight trait.

- **Internet-free Communication:** Unlike majority of MCC solutions that need to connect to distant cloud datacenters through Internet, this framework establishes the connection with the MNO or nearby MSPs without need to pass through the risky insecure channel of the Internet. Apart from security hazards of communicating via Internet, utilizing the Internet services to accessing cloud resources originates long WAN latency and levies billing cost of data plan on the users that decrease usability of MCC solutions. Therefore, the Internet-free communication aspect of our proposed MCC framework enhances adoption and deployment of computing empowerment of mobile devices in MCC.

- **Centralized Architecture:** We reduce system complexity and security risk of communicating with insecure MSPs by employing the MNOs as TSG to control the authenticity and reliability of MSPs at registration time and throughout their life time. Such improved trust is achieved when TSG enforces MSPs to prove their identity at the time of registration to discourage them from possible attack to MSC. Because of historical trust that MNOs could build over the years, users do not hesitate (or at least hesitate less) to utilize MNO services to empower their mobile devices and save their local resources.

- **Short WAN Latency:** Leveraging cloud of proximate mobile devices as MSP significantly reduces the communication and WAN latency of accessing remote resources. Mobile devices are the most proximate ubiquitous computing devices to MSCs. Hence, utilizing such pervasive mobile devices decreases the WAN latency and improves RMA execution efficiency.

- **Low Communication Overhead:** In our proposal, we neither partition the intensive codes nor send the code to the TSG or MSP. The intensive codes are already

available in the TSG and all registered MSPs. Hence, there is no need to migrate the codes to the MSP at execution time. Considering large number of mobile devices and insatiable demand of mobile users to perform RMAs in their resource-constraint mobile devices, the communication overhead of transferring codes to MSPs in wireless network would be highly considerable in near future. Therefore, omitting overhead of transmitting intensive codes to the MSPs remarkably decreases the communication overhead of computing networks.

- **Mobile Service Provisioning:** The concept of mobile service provisioning is another significant feature of this proposal. Traditional service-based systems entertain three main roles of service development and provisioning, service consuming, and service brokering. However, designing our framework based on such philosophy complicates service provisioning for mobile device owners because of mandatory IT skill acquisition.

  We address this challenge by separating service development from service provisioning. Therefore, in our model, four major roles are designed, namely service development, service provisioning, service consuming, and service brokering. Consequently, our system has four major building blocks, including service developer (skilled service programmer), mobile service consumer, mobile service provider (mobile device owner), and service broker (TSG). Using our deployed design philosophy, owners of mobile devices can simply browse the TSG's database, select the service to provision on their mobile devices, install the code and perform resource provisioning without need to acquire programming and IT skills. Thus, it motivates individual mobile owners to leverage this framework to participate in cloud-based mobile service provisioning and earn credit (financial or non-financial).

- **Green Mobile Computing:** One of the most important features of this framework is its approach toward green mobile computing. This framework enables owners of off-the-shelve or faulty-yet-functioning mobile devices to share computing resources of their devices either for free or in exchange of certain credit (either reputation or finance) if billing system is incorporated into the system. Considering huge number of rapidly antiquated mobile devices, this approach can save the environment by more efficient utilization of old devices. Due to time-consuming nature of maintenance operation of such old mobile devices, it is an effective way to leverage wisdom of individual mobile owners in MCC and encourage them to maintain their devices for service provisioning.

## 7.4 International Scholarly Publications

The following is the list of accepted/published articles related to research undertaken in this thesis.

**Authored Publications**

1. S.Abolfazli, Z.Sanaei, E.Ahmed, A.Gani, R.Buyya, Cloud-based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges, **IEEE Communications Surveys & Tutorials** (World $1^{st}$ ranked ISI CS journal, Q1, IF=6.31), Vol 16, No.1 pp. 337-368, Feb 2014, DOI:10.1109/SURV.2013.070813.00285.

2. S.Abolfazli, Z.Sanaei, A.Gani, F.Xia, L. T. Yang, Rich Mobile Applications: Genesis, Taxonomy, and Open Issues, **Journal of Network and Computer Applications** (Q1, IF=1.46), Vol. 40, pp. 345-362, April 2014, 10.1016/j.jnca.2013.09.009.

3. S.Abolfazli, Z.Sanaei, A.Gani, An Experimental Analysis on Cloud-based Mobile Augmentation in Mobile Cloud Computing, **IEEE Transactions on Consumer Electronics** (Q2, IF=1.087), Vol. 40, Issue 1, pp. 146-154, 2014.

4. S. Abolfazli, Z. Sanaei, A. Gani, F. Xia, W. Lin, RMCC: RESTful Mobile Cloud Computing Framework for Exploiting Adjacent Service-based Mobile Cloudlets, **IEEE CloudCom14**, Accepted, Singapore, 2014.

5. S. Abolfazli, Z. Sanaei, M. Shiraz, A. Gani, MOMCC: Market-Oriented Architecture for Mobile Cloud Computing Based on Service Oriented Architecture , **IEEE Workshop on Mobile Cloud Computing (MobiCC 2012)**, Beijing, China, 2012, pp. 8-16.

6. S. Abolfazli, Z. Sanaei, A. Gani, Mobile Cloud Computing: A Review on Smartphone Augmentation, **The 1st International Conference on Computing, Information Systems and Communications (CISCO' 12)**, Singapore, 2012, pp. 199-204.

7. S. Abolfazli, Z.Sanaei, M.H. Sanaei, A.Gani, Mobile Cloud Computing: The-state-of-the-art, Challenges, and Future Researches, **Chapter in Encyclopedia of Cloud Computing, Wiley, 2014**, Feb 2014.

8. S. Abolfazli, Z. Sanaei, A. Gani, A Lightweight Mobile Cloud Computing Platform for Resource-intensive Mobile Applications, **University Malaya Research Conference (UMRC'13)**, Kuala Lumpur, Malaysia, Nov 2013.

**Co-Authored Publications**

9. Z.Sanaei, S.Abolfazli, A.Gani, R. Buyya, Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges, **IEEE Communications Surveys & Tutorials** (Q1, IF=6.3), Vol 16, No.1 pp. 369-392, 10.1109/SURV.2013.050113.00090, Feb 2014.

10. E. Ahmad, A. Gani, S. Abolfazli, J. Y. Liu, S. U. Khan, Channel Assignment Algorithms in Cognitive Radio Networks: Taxonomy, Open Issues, and Challenges, **IEEE Communications Surveys & Tutorials**, In Press, Oct 2014.

11. Z.Sanaei, S.Abolfazli, T. Alizadeh, F. Xia, Hybrid Pervasive Mobile Cloud Computing: Toward Enhancing Invisibility, **Information**, Vol 16, No 11, pp. 8145-8156, Nov 2013.

12. M. Shiraz, S. Abolfazli, Z.Sanaei, A.Gani, A study on virtual machine deployment for application outsourcing in mobile cloud computing, **The Journal of Supercomputing** (Q2), Vol 62, No. 3, 2012, DOI 10.1007/s11227-012-0846-y.

13. Z. Sanaei, S. Abolfazli, A. Gani, M. Shiraz, SAMI: Service-Based Arbitrated Multi-Tier Infrastructure for Mobile Cloud Computing, IEEE Workshop on Mobile Cloud Computing **(MobiCC 2012)**, China, 2012, PP. 14-19.

14. Z. Sanaei, S. Abolfazli, A. Gani, R. H. Khokhar, Tripod of Requirements in Horizontal Heterogeneous Mobile Cloud Computing, The 1st International Conference on Computing, Information Systems and Communications **(CISCO' 12)**, 2012, Singapore, PP.217-222.

## 7.5 Future Works

Although performance evaluation and validation in this study advocate feasibility and significant performance gain in deployment of our lightweight MCC framework, further

efforts in future can enhance different aspects of the framework leading to it success and adoption. One possible future work is to develop a sub-system than can help application developers to locate available services and automate servide discovery. Although there are huge number of available web services, discovering desired services that can be employed in mobile devices requires extra efforts by the mobile application developer.

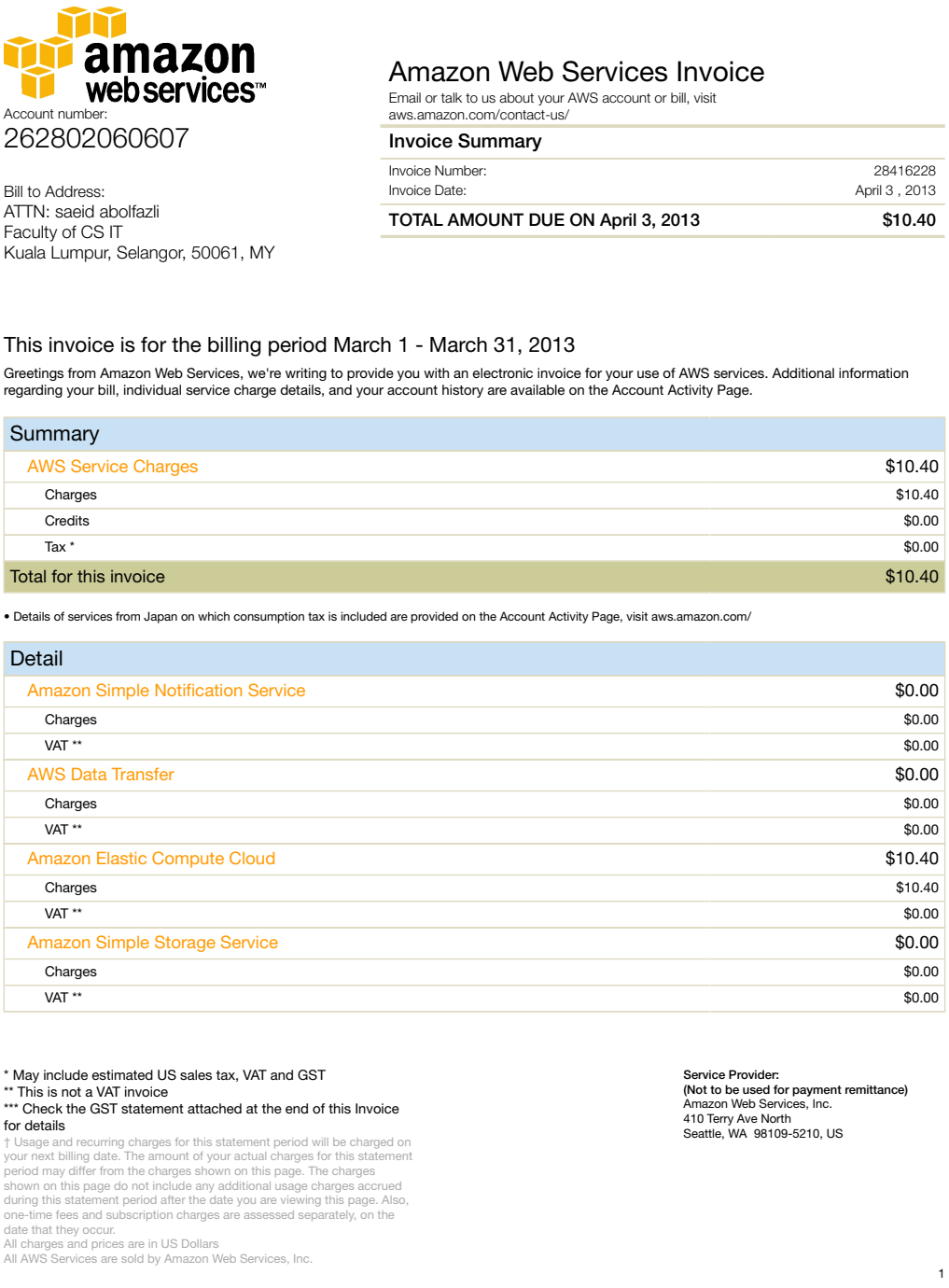
Additionally, despite the significant mitigation of latency between MSC and MSPs in this framework, communication latency of transmitting data over the network, especially wireless network is still noticeable and consumes remarkable portion of the time and energy in PMC mode. Future works can enhance communication latency. This overhead is increasing as the data intensity of workloads increases. Also, incorporating a decision making engine that can leverage heterogeneous network technologies such as Bluetooth, wi-fi, and 3G can enhance flexibility and adaptability of our framework.

Furthermore, considering both MSC and MSPs' mobility, mobility management component is of a crucial necessity in future. When mobile devices move, there is chance that MSP gets out of MSC's coverage. Thus, the MSC requires repeating the augmentation process to finds another MSP in vicinity to perform the lost computation. Similarly, the computing resources of MSP are wasted by the orphan processes. Results of the orphan processes cannot be delivered to the MSC due to inaccessibility and unavailability issues. Moreover, security and privacy in the system demand future efforts, though the TSG avoid hosting of services by unauthorized mobile host.

Futuristic mobile billing systems can remarkably improve the system deployment and adoption. An efficient billing system is expected to offer encouraging and lucrative incentive for service developers and MSPs beside an economy and affordable billing method that encourages MSCs to consume services of our framework delivered via MSPs.

# Appendices

We utilize the cloud services of Amazon EC2 Web Services to perform the real benchmarking experiments of our problem analysis. The following are the invoices issued by Amazon Web Services from March to June 2013 period.



amazon web services™

Account number:
262802060607

Bill to Address:
ATTN: saeid abolfazli
Faculty of CS IT
Kuala Lumpur, Selangor, 50061, MY

Amazon Web Services Invoice

Email or talk to us about your AWS account or bill, visit aws.amazon.com/contact-us/

**Invoice Summary**

| | |
|---|---|
| Invoice Number: | 28416228 |
| Invoice Date: | April 3 , 2013 |
| **TOTAL AMOUNT DUE ON April 3, 2013** | **$10.40** |

This invoice is for the billing period March 1 - March 31, 2013

Greetings from Amazon Web Services, we're writing to provide you with an electronic invoice for your use of AWS services. Additional information regarding your bill, individual service charge details, and your account history are available on the Account Activity Page.

| Summary | |
|---|---|
| AWS Service Charges | $10.40 |
| Charges | $10.40 |
| Credits | $0.00 |
| Tax * | $0.00 |
| Total for this invoice | $10.40 |

• Details of services from Japan on which consumption tax is included are provided on the Account Activity Page, visit aws.amazon.com/

| Detail | |
|---|---|
| Amazon Simple Notification Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| AWS Data Transfer | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| Amazon Elastic Compute Cloud | $10.40 |
| Charges | $10.40 |
| VAT ** | $0.00 |
| Amazon Simple Storage Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |

* May include estimated US sales tax, VAT and GST
** This is not a VAT invoice
*** Check the GST statement attached at the end of this Invoice for details
† Usage and recurring charges for this statement period will be charged on your next billing date. The amount of your actual charges for this statement period may differ from the charges shown on this page. The charges shown on this page do not include any additional usage charges accrued during this statement period after the date you are viewing this page. Also, one-time fees and subscription charges are assessed separately, on the date that they occur.
All charges and prices are in US Dollars
All AWS Services are sold by Amazon Web Services, Inc.

**Service Provider:**
**(Not to be used for payment remittance)**
Amazon Web Services, Inc.
410 Terry Ave North
Seattle, WA 98109-5210, US

1

Figure .1: Invoice issued by Amazon Web Services for utilized cloud services in March 2013

Account number:
262802060607

Bill to Address:
ATTN: saeid abolfazli
Faculty of CS IT
Kuala Lumpur, Selangor, 50061, MY

# Amazon Web Services Invoice

Email or talk to us about your AWS account or bill, visit aws.amazon.com/contact-us/

## Invoice Summary

| | |
|---|---|
| Invoice Number: | 29016807 |
| Invoice Date: | May 3 , 2013 |
| **TOTAL AMOUNT DUE ON May 3, 2013** | **$7.08** |

## This invoice is for the billing period April 1 - April 30, 2013

Greetings from Amazon Web Services, we're writing to provide you with an electronic invoice for your use of AWS services. Additional information regarding your bill, individual service charge details, and your account history are available on the Account Activity Page.

| Summary | |
|---|---|
| AWS Service Charges | $7.08 |
| Charges | $7.08 |
| Credits | $0.00 |
| Tax * | $0.00 |
| Total for this invoice | $7.08 |

• Details of services from Japan on which consumption tax is included are provided on the Account Activity Page, visit aws.amazon.com/

| Detail | |
|---|---|
| Amazon Simple Notification Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| AWS Data Transfer | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| Amazon Elastic Compute Cloud | $7.08 |
| Charges | $7.08 |
| VAT ** | $0.00 |
| Amazon Simple Storage Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |

* May include estimated US sales tax, VAT and GST
** This is not a VAT invoice
*** Check the GST statement attached at the end of this Invoice for details
† Usage and recurring charges for this statement period will be charged on your next billing date. The amount of your actual charges for this statement period may differ from the charges shown on this page. The charges shown on this page do not include any additional usage charges accrued during this statement period after the date you are viewing this page. Also, one-time fees and subscription charges are assessed separately, on the date that they occur.
All charges and prices are in US Dollars
All AWS Services are sold by Amazon Web Services, Inc.

**Service Provider:**
**(Not to be used for payment remittance)**
Amazon Web Services, Inc.
410 Terry Ave North
Seattle, WA 98109-5210, US

1

Figure .2: Invoice issued by Amazon Web Services for utilized cloud services in April 2013

Account number:
262802060607

Bill to Address:
ATTN: saeid abolfazli
Faculty of CS IT
Kuala Lumpur, Selangor, 50061, MY

# Amazon Web Services Invoice

Email or talk to us about your AWS account or bill, visit aws.amazon.com/contact-us/

## Invoice Summary

| | |
|---|---|
| Invoice Number: | 29911774 |
| Invoice Date: | June 3 , 2013 |
| **TOTAL AMOUNT DUE ON June 3 , 2013** | **$3.34** |

## This invoice is for the billing period May 1 - May 31 , 2013

Greetings from Amazon Web Services, we're writing to provide you with an electronic invoice for your use of AWS services. Additional information regarding your bill, individual service charge details, and your account history are available on the Account Activity Page.

| Summary | |
|---|---|
| AWS Service Charges | $3.34 |
| Charges | $3.34 |
| Credits | $0.00 |
| Tax * | $0.00 |
| Total for this invoice | $3.34 |

• Details of services from Japan on which consumption tax is included are provided on the Account Activity Page, visit aws.amazon.com/

| Detail | |
|---|---|
| Amazon Simple Notification Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| AWS Data Transfer | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| Amazon Elastic Compute Cloud | $3.34 |
| Charges | $3.34 |
| VAT ** | $0.00 |
| Amazon Simple Storage Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |

* May include estimated US sales tax, VAT and GST
** This is not a VAT invoice
*** Check the GST statement attached at the end of this Invoice for details
† Usage and recurring charges for this statement period will be charged on your next billing date. The amount of your actual charges for this statement period may differ from the charges shown on this page. The charges shown on this page do not include any additional usage charges accrued during this statement period after the date you are viewing this page. Also, one-time fees and subscription charges are assessed separately, on the date that they occur.
All charges and prices are in US Dollars
All AWS Services are sold by Amazon Web Services, Inc.

**Service Provider:**
**(Not to be used for payment remittance)**
Amazon Web Services, Inc.
410 Terry Ave North
Seattle, WA 98109-5210, US

1

Figure .3: Invoice issued by Amazon Web Services for utilized cloud services in May 2013

Account number:
262802060607

Bill to Address:
ATTN: saeid abolfazli
Faculty of CS IT
Kuala Lumpur, Selangor, 50061, MY

# Amazon Web Services Invoice

Email or talk to us about your AWS account or bill, visit aws.amazon.com/contact-us/

## Invoice Summary

| | |
|---|---|
| Invoice Number: | 30194110 |
| Invoice Date: | July 3 , 2013 |
| **TOTAL AMOUNT DUE ON July 3 , 2013** | **$3.32** |

## This invoice is for the billing period June 1 - June 30 , 2013

Greetings from Amazon Web Services, we're writing to provide you with an electronic invoice for your use of AWS services. Additional information regarding your bill, individual service charge details, and your account history are available on the Account Activity Page.

| Summary | |
|---|---|
| AWS Service Charges | $3.32 |
| Charges | $3.32 |
| Credits | $0.00 |
| Tax * | $0.00 |
| Total for this invoice | $3.32 |

• Details of services from Japan on which consumption tax is included are provided on the Account Activity Page, visit aws.amazon.com/

| Detail | |
|---|---|
| AWS Data Transfer | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| Amazon Simple Notification Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |
| Amazon Elastic Compute Cloud | $3.32 |
| Charges | $3.32 |
| VAT ** | $0.00 |
| Amazon Simple Storage Service | $0.00 |
| Charges | $0.00 |
| VAT ** | $0.00 |

* May include estimated US sales tax, VAT and GST
** This is not a VAT invoice
*** Check the GST statement attached at the end of this Invoice for details
† Usage and recurring charges for this statement period will be charged on your next billing date. The amount of your actual charges for this statement period may differ from the charges shown on this page. The charges shown on this page do not include any additional usage charges accrued during this statement period after the date you are viewing this page. Also, one-time fees and subscription charges are assessed separately, on the date that they occur.
All charges and prices are in US Dollars
All AWS Services are sold by Amazon Web Services, Inc.

**Service Provider:**
**(Not to be used for payment remittance)**
Amazon Web Services, Inc.
410 Terry Ave North
Seattle, WA 98109-5210, US

1

Figure .4: Invoice issued by Amazon Web Services for utilized cloud services in June 2013

# REFERENCES


Aho, A., & Ullman, J. (1992). *Foundations of Computer Science*. W. H. Freeman.

Akyildiz, I. F., Jiang, X., & Mohanty, S. (2004). A survey of mobility management in next-generation all-IP-based wireless systems. *IEEE Wireless Communications*, *11*(4), 16–28.

Albanesius, C. (2011). *Smartphone Shipments Surpass PC Shipments for First Time. What's Next?* Retrieved 22 Sep 2014, from http://www.pcmag.com/article2/0,2817,2379665,00.asp

A.Manjunatha, A.Ranabahu, A.Sheth, & K.Thirunarayan. (2010, November). Power of Clouds In Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development. In *Proc. ieee cloudcom'10* (pp. 496–503). Indiana, USA.

Badidi, E., & Taleb, I. (2011). Towards a cloud-based framework for context management. In *Proc. ieee iit'11* (pp. 35–40). Abu Dhabi, United Arab Emirates.

Bahl, P., Han, R. Y., Li, L. E., & Satyanarayanan, M. (2012). Advancing the state of mobile cloud computing. In *Proc. acm mcs '12* (pp. 21–28). Helsinki, Finland.

Bratus, S., Masone, C., & Smith, S. W. (2008). Why Do Street-Smart People Do Stupid Things Online? *IEEE Security & Privacy*, *6*(3), 71–74.

Caballe, S., Xhafa, F., & Barolli, L. (2010). Using mobile devices to support online collaborative learning. *Mobile Information Systems*, *6*(1, SI), 27–47.

Cachin, C., & Schunter, M. (2011, December). A cloud you can trust. *IEEE Spectrum*, *48*(12), 28–51.

Chang, H. B., Kwon, H. J., & Kang, J. G. (2010). The design and implementation of tamper resistance for mobile game service. *Mobile Information Systems*, *6*(1, SI), 85–105.

Christensen, J. H. (2009). Using RESTful web-services and cloud computing to create next generation mobile applications. *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, 627–634. doi: 10.1145/1639950.1639958

Chun, B. G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011). CloneCloud: Elastic execution between mobile device and cloud. In *Proc. acm eurosys'11* (pp. 301–314). Salzburg, Austria.

*Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018.* (2013). Cisco. Retrieved from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., ... Warfield, A. (2005). Live migration of virtual machines. In *Proc. usenix nsdi' 05* (pp. 273–286). Boston, MA, USA.

Clark, J. (2013, February). *Microsoft secure Azure Storage goes down Wordwide.* Retrieved 22/9/2014, from http://www.theregister.co.uk/2013/02/22/azure_problem_that_should_never_happen_ever/

Cook, T., Campbell, D., & Day, A. (1979). *Quasi-experimentation: Design & analysis issues for field settings*. Retrieved from http://dickyh.staff.ugm.ac.id/wp/wp-content/uploads/2009/ringkasanbukuquasi-experimentakhir.pdf

Cordeschi, N., Shojafar, M., & Baccarelli, E. (2013). Energy-saving self-configuring networked data centers. *Computer Networks*, *57*(17), 3479–3491.

Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). MAUI:

making smartphones last longer with code offload. In *Proc. acm mobisys'10* (pp. 49–62). San Francisco, USA.

Curnow, Harold J., & Wichmann, B. A. (1976). A synthetic benchmark. *The Computer Journal*, *19*(1), 43–49.

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, *51*(1), 107–113.

Deitel, P., & Deitel, H. (2008). *Ajax, rich internet applications, and web development for programmers* (Deitel dev ed.). Prentice Hall Press.

Di Francesco, M. (2012, November). Energy consumption of remote desktop access on mobile devices: An experimental study. In *2012 ieee 1st international conference on cloud networking (cloudnet)* (pp. 105–110). Paris: Ieee.

Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., & Vakali, A. (2009, September). Cloud Computing: Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing*, *13*(5), 10–13. doi: 10.1109/MIC.2009.103

Enck, W., Ongtang, M., & McDaniel, P. (2011). On lightweight mobile phone application certification. In *Proc. ieee ccs'11* (pp. 235–245). Milan, Italy.

Erl, T. (2008). *Soa: principles of service design*. Retrieved from `http://www.weibnc.com/wp-content/uploads/brkpdfs/SOA-Principles-of-Service-Design-by-Thomas-ErlGood-Book.pdf`

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (PhD Thesis). University California, Irvin.

Flinn, J., & Satyanarayanan, M. (1999). Energy-aware adaptation for mobile applications. In *Proc. acm sosp' 99* (Vol. 33, pp. 48–63).

Gantz, J. F., Chute, C., Manfrediz, A., Minton, S., Reinsel, D., Schlichting, W., & Toncheva, A. (2008). *The Diverse and Exploding Digital Universe: An Updated Forecast of Worldwide Information Growth Through 2011.* White Paper. International Data Corporation.

Garriss, S., Cáceres, R., Berger, S., Sailer, R., van Doorn, L., & Zhang, X. (2008). Trustworthy and personalized computing on public kiosks. In *Proc. acm mobisys '08* (pp. 199–210). Breckenridge, CO, USA.

*Gartner Identifies the Top 10 Strategic Technology Trends for 2015.* (2014). Gartner. Retrieved 30 OCT 2014, from `http://www.gartner.com/newsroom/id/2867917`

Gu, Y., March, V., & Lee, B. S. (2012). GMoCA: Green mobile cloud applications. In *Proc. ieee greens'12* (pp. 15–20). Zurich, Switzerland. doi: 10.1109/GREENS.2012.6224265

Guinard, D., Trifa, V., & Wilde, E. (2010, November). A resource oriented architecture for the Web of Things. In *2010 internet of things (iot)* (pp. 1–8). IEEE. Retrieved from `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5678452` doi: 10.1109/IOT.2010.5678452

Guirguis, M., Ogden, R., Song, Z., Thapa, S., & Gu, Q. (2011, December). Can You Help Me Run These Code Segments on Your Mobile Device? In *Proc. ieee globecom '11* (pp. 1–5). Houston, Texas, USA. doi: 10.1109/GLOCOM.2011.6134213

Guo, Y., Zhang, L., Kong, J., Sun, J., Feng, T., & Chen, X. (2011). Jupiter: transparent augmentation of smartphone capabilities through cloud computing. In *Proc. acm mobiheld '11* (p. 2). Cascais, Portugal.

Hariharan, K. (2008). Best Practices: Extending Enterprise Applications to Mobile Devices. *The Architec-*

*ture Journal, Microsoft Architecture Center*, *14*.

Hazmi, A., Rinne, J., & Valkama, M. (2012, December). Feasibility study of IEEE 801.11 ah radio technology for IoT and M2M use cases. In *2012 ieee globecom workshops* (pp. 1687–1692). IEEE. doi: 10.1109/GLOCOMW.2012.6477839

Huerta-Canepa, G., & Lee, D. (2010). A Virtual Cloud Computing Provider for Mobile Devices. In *Proc. acm mcs'10: Social networks and beyond* (pp. 1–5). San Francisco, USA.

Hung, S.-H., Shih, C.-S., Shieh, J.-P., Lee, C.-P., & Huang, Y.-H. (2011). Executing mobile applications on the cloud: Framework and issues. *Computers and Mathematics with Applications*, *63*(2), 573–587.

Jain, R. (2008). *The art of computer systems performance analysis*. Wiley.

Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014). Hybrid Job Scheduling Algorithm for Cloud Computing Environment. In *Int'l conf. innovations in bio-inspired computing and applications* (pp. 43–52).

Kamara, S., & Lauter, K. (2010). Cryptographic cloud storage. *Financial Cryptography and Data Security*, 136–149.

Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2012). The Smartphone and the Cloud: Power to the User. In *Proc. mobicase '12* (pp. 342–348). California, USA.

Khan, A. N., Mat Kiah, M., Khan, S. U., & Madani, S. A. (2013). Towards secure mobile cloud computing: A survey. *Future Generation Computer Systems*, *29*(5), 1278–1299. doi: 10.1016/j.future.2012.08.003

Kim, H., & Parashar, M. (2011). CometCloud: An Autonomic Cloud Engine. *Cloud Computing: Principles and Paradigms*, 275–297.

Kim, R. Y., & Mohanty, S. (2010). Advanced power management techniques in next-generation wireless networks [Topics in Wireless Communications]. *IEEE Communications Magazine*, *48*(5), 94–102.

Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012). ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proc. ieee infocom '12* (pp. 945–953). Orlando, USA.

Kremer, U., Hicks, J., & Rehg, J. (2003). A Compilation Framework for Power and Energy Management on Mobile Computers. In *Proceedings of the 14th international conference on languages and compilers for parallel computing* (pp. 115–131). Retrieved from `http://dl.acm.org/citation.cfm?id=1769331.1769339`

Kristensen, M. D. (2010). Scavenger: Transparent development of efficient cyber foraging applications. In *Proc. percom'10* (pp. 217–226). Mannheim, Germany.

Kumar, K., & Lu, Y. H. (2010). Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? *IEEE Computer*, *43*(4), 51–56.

Lane, N. D., Miluzzo, E., Hong, L., Peebles, D., Choudhury, T., & Campbell, A. T. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine*, *48*(9), 140–150.

Lawton, G. (2008). New Ways to Build Rich Internet Applications. *Computer*, *41*(8), 10–12.

Lawton, G. (2012). Cloud Streaming Brings Video to Mobile Devices. *IEEE Computer*, *45*(2), 14–16.

Li, Z., Wang, C., & Xu, R. (2001). Computation offloading to save energy on handheld devices: a partition scheme. *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, 238–246. Retrieved from `http://dl.acm.org/citation.cfm?id=502257`

doi: 10.1145/502217.502257

Lu, Y., Li, S. P., & Shen, H. F. (2011). Virtualized Screen: A Third Element for Cloud-Mobile Convergence. *IEEE Multimedia*, *18*(2), 4–11.

Lukowicz, P., Pentland, S., & Ferscha, A. (2012, January). From Context Awareness to Socially Aware Computing. *Pervasive Computing*, *11*(1), 32–41.

Ma, R. K. K., & Wang, C. L. (2012). Lightweight Application-Level Task Migration for Mobile Cloud Computing. In *Proc. ieee advanced information networking and applications (aina'12)* (pp. 550–557). Fukuoka.

Makris, P., Skoutas, D. N., & Skianis, C. (2013). A Survey on Context-Aware Mobile and Wireless Networking: On Networking and Computing Environments' Integration. *IEEE Communications Surveys & Tutorials*, *15*(1), 362–386. doi: 10.1109/SURV.2012.040912.00180

March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., & Lee, B. S. (2011). $\mu$Cloud: Towards a New Paradigm of Rich Mobile Applications. In *Proc. 8th int'l conf. mobile web information systems {(MobiWIS'11)}* (Vol. 5, pp. 618–624). Ontario, Canada.

Marinelli, E. E. (2009). *Hyrax: cloud computing on mobile devices using MapReduce* (Master Thesis). Computer Science Department, Carnegie Mellon University.

Marquardt, P., Verma, A., Carter, H., & Traynor, P. (2011). (sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proc. acm ccs '11* (pp. 551–562). Chicago, IL, USA.

Mather, T., Kumaraswamy, S., & Latif, S. (2009). *Cloud security and privacy: an enterprise perspective on risks and compliance*. O'Reilly Media, Incorporated.

Mei, C., Taylor, D., Wang, C., Chandra, A., & Weissman, J. (2012, June). Sharing-Aware Cloud-Based Mobile Outsourcing. In *Proc. ieee cloud '12* (pp. 408–415). Hawaii, USA.

Mizouni, R., Serhani, M. A., Dssouli, R., Benharref, A., & Taleb, I. (2011, September). Performance Evaluation of Mobile Web Services. In *Ieee 9th european conf. web services* (pp. 184–191). Lugano.

MONACO, A. (2012). *Beware: Free Apps Might Prove Costly.* Retrieved from `http://theinstitute.ieee.org/technology-focus/technology-topic/beware-free-apps-might-prove-costly`

Natchetoi, Y., Kaufman, V., & Shapiro, A. (2008). Service-Oriented Architecture for Mobile Applications. In *Proc. acm sam '08* (pp. 27–32). Leipzig, Germany.

Norman, D., & Draper, S. (1986). *User centered system design; new perspectives on human-computer interaction*. L. Erlbaum Associates Inc.

Othman, M., & Hailes, S. (1998). Power conservation strategy for mobile computers using load sharing. *ACM SIGMOBILE Mobile Computing and Communications Review*, *2*(1), 44–51.

Owens, K. (2011). *Securing Virtual Compute Infrastructure in the Cloud.* Whitepaper. Retrieved from `http://www.knowledgestorm.com/sol_summary_5152030.asp`

Perrucci, G. P., Fitzek, F. H. P., & Widmer, J. (2011). Survey on Energy Consumption Entities on the Smartphone Platform. In *Proc. ieee vtc spring '11* (pp. 1–6). Budapest, Hungary.

Pickard, W. F., & Abbott, D. (2012). Addressing the Intermittency Challenge: Massive Energy Storage in a Sustainable Future [Scanning the Issue]. *Proceedings of the IEEE*, *100*(2), 317–321.

*PowerTutor:A Power Monitor for Android-Based Mobile Platforms.* (n.d.). Retrieved from `http://ziyang.eecs.umich.edu/projects/powertutor/`

Prosper Mobile Insights. (2011). *Smartphone/Tablet User Survey.* Retrieved 22-Sep-2014, from http://prospermobileinsights.com/Default.aspx?pg=19

Rahimi, M., Venkatasubramanian, N., Mehrotra, S., & Vasilakos, V. (2012). MAPCloud: mobile applications on an elastic and scalable 2-tier cloud architecture. In *Proc. ieee/acm ucc'12.* Chicago, Illionis , USA.

Ramaswamy, R., Ning, W., & T.Wolf. (2004). Characterizing network processing delay. In *Ieee globecom '04* (pp. 1629–1634). Texas, USA.

Ranabahu, A. H., Maximilien, E. M., Sheth, A. P., & Thirunarayan, K. (2011). A domain specific language for enterprise grade cloud-mobile hybrid applications. In *Proc. acm co-located workshops on dsm'11, tmc'11, agere'11, aoopes'11, neat'11, & vmil'11* (pp. 77–84). Portland, Oregon, USA.

R.Avro, S. (2009). *Wireless Electricity is Real and Can Change the World* (No. 10-12-2011). Consumer Energy Report. Retrieved 22-Sep-2014, from http://www.consumerenergyreport.com/2009/01/15/wireless-electricity-is-real-and-can-change-the-world/

Ren, K., Wang, C., & Wang, Q. (2012). Security challenges for the public cloud. *IEEE Internet Computing*, *16*, 69–73. doi: 10.1109/MIC.2012.14

Richardson, L., & Ruby, S. (2007). *RESTful Web Services*. O'Reilly Media.

R.Kemp, Palmer, N., Kielmann, T., & Bal, H. (2010). Cuckoo: a Computation Offloading Framework for Smartphones. In *Proc. springer mobicase '10* (pp. 59–79). Santa Clara, CA, USA.

Robinson, S. (2009). *Cellphone Energy Gap: Desperately Seeking Solutions* (No. 30-11-2011). Strategy Analytics. Retrieved 22-Sep-2014, from http://strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=4645

Satyanarayanan, M. (2005). Avoiding Dead Batteries. *IEEE Pervasive Computing*, *4*(1), 2–3.

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, *8*(4), 14–23.

security, P. (2013). *Panda Global Business Protection on-premise security solutions.* Retrieved 22-Sep-2014, from http://www.pandasecurity.com/usa/enterprise/security-solutions-on-premise/

Sharifi, M., Kafaie, S., & Kashefi, O. (2011). A Survey and Taxonomy of Cyber Foraging of Mobile Devices. *IEEE Communications Surveys & Tutorials*, *PP*(99), 1–12.

Shen, S., & Blau, B. (2012, August). *Market Trends: Mobile App Stores, Worldwide, 2012.* Retrieved 22-Sep-2014, from http://www.gartner.com/id=2126015

Shiraz, M., Gani, A., Hafeez Khokhar, R., & Buyya, R. (2012, November). A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing. *IEEE Communications Surveys & Tutorials*, *15*(3), 1294 – 1313.

Siegal, J. (2014). *Despite iPhone 6 hype, Android continues to dominate iOS market share.* BGR. Retrieved from http://bgr.com/2014/07/01/android-market-share-2014/

Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., & Heinzelman, W. (2012). Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *Proc. ieee iscc '12* (pp. 59–66). Cappadocia, Turkey.

Starner, T., Kirsch, D., & Assefa, S. (1997). The Locust Swarm: An environmentally-powered, networkless location and messaging system. In *Proc. ieee iswc' 97* (pp. 169–170). Boston, MA, USA.

Takabi, H., Joshi, J. B. D., & Ahn, G. J. (2010). Security and Privacy Challenges in Cloud Computing

Environments. *IEEE Security & Privacy*, *8*(6), 24–31.

Tolia, N., Andersen, D. G., & Satyanarayanan, M. (2006). Quantifying interactive user experience on thin clients. *IEEE Computer*, *39*(3), 46–52.

*Top Technology Trends for 2014.* (2014). Retrieved from `http://www.computer.org/portal/web/membership/Top-10-Tech-Trends-in-2014`

Vallina-Rodriguez, N., & Crowcroft, J. (2013). Energy Management Techniques in Modern Mobile Handsets. *IEEE Communications Surveys & Tutorials*, *15*(1), 179–198.

van Deursen, A., Klint, P., & Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices*, *35*(6), 26–36.

Verbelen, T., Simoens, P., De Turck, F., & Dhoedt, B. (2012, November). AIOLOS: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software*, *85*(11), 2629–2639. doi: 10.1016/j.jss.2012.06.011

Wang, C., Wang, Q., Ren, K., & Lou, W. (2009). Ensuring data storage security in cloud computing. In *Proc. ieee iwqos '09* (pp. 1–9). Charleston, South Carolina, USA.

White, T. (2012). *Hadoop: The definitive guide*. O'Reilly Media.

Whitten, A. (2004). *Making security usable* (Unpublished doctoral dissertation). Princeton University.

Xia, F., Ding, F., Li, J., Kong, X., Yang, L. T., & Ma, J. (2013, October). Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. *Information Systems Frontiers*, *16*(1), 95–111. doi: 10.1007/s10796-013-9458-1

Youseff, L., Wolski, R., Gorda, B., & Krintz, C. (2006). Paravirtualization for HPC Systems. In G. Min, B. Martino, L. Yang, M. Guo, & G. Ruenger (Eds.), *Frontiers of high performance computing and networking-ispa 2006 workshops* (Vol. 4331, pp. 474–486).

Zhang, X. W., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011a). Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing. *Mobile Networks & Applications*, *16*(3), 270–284.

Zhang, X. W., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011b). Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing. *Mobile Networks & Applications*, *16*(3), 270–284.

Zheng, W., Xu, P., Huang, X., & Wu, N. (2010). Design a cloud storage platform for pervasive computing environments. *Cluster Computing*, *13*(2), 141–151.

# LIST OF SYMBOLS AND ACRONYMS

| | |
|---|---|
| 4G | 4th Generation. |
| ACH | Asynchronous Communication Handler. |
| CMA | Cloud-based Mobile Augmentation. |
| CPU | Central Processing Unit. |
| CRC | Cyclic Redundancy Check. |
| CRM | Customer Relation Management. |
| CWnd | Congestion Window. |
| DIC | Distant Immobile Cloud. |
| DSL | Domain Specific Language. |
| DVMS | Dynamic VM Synthesis. |
| GPU | Graphical Processing Unit. |
| GUI | Graphical User Interface. |
| IMSI | International Mobile Subscriber Identity. |
| IP | Internet Protocol. |
| LAI | Location Area Identifier. |
| LCD | Liquid Crystal Display. |
| MCC | Mobile Cloud Computing. |
| ME2 | Mobile Empowering Engine. |
| MNO | Mobile Network Operators. |
| MSC | Mobile Service Consumer. |
| MSP | Mobile Service Provider. |
| MTU | Maximum Transmission Unit. |
| OCR | Optical Character Recognition. |
| QoS | Quality of Service. |
| RAM | Random Access Memory. |
| REST | Representational State Transfer. |
| RISC | Reduced Instruction Set Computing. |
| RMA | Resource-intensive Mobile Application. |

| | |
|---|---|
| ROA | Resource-Oriented Architecture. |
| SLA | Service Level Agreement. |
| SOAP | Simple Object Access Protocol. |
| SPDE | Service Provider Discovery Engine. |
| TCP | Transmission Control Protocol. |
| TSG | Trusted Service Governor. |
| UDDI | Universal Description Discovery and Integrity. |
| URI | Unified Resource Identifier. |
| VLR | Visitor Location Register. |
| VM | Virtual Machine. |
| VMM | Virtual Machine Manager. |
| VPU | Virtual Processing Unit. |
| WAN | Wide Area Network. |
| Wi-Fi | Wireless Fidelity. |
| WLAN | Wireless Local Area Network. |